Universidad Autónoma de Madrid
Escuela politécnica superior

Máster Oficial en Bioinformática y Biología Computacional

Trabajo Fin de Máster

NGSCAT2. AN IMPROVED TOOL TO ASSESS THE EFFICIENCY OF TARGETED ENRICHMENT SEQUENCING

Autor: Alejandro García Sánchez
Tutor: Javier Pérez Florido
Ponente: Enrique Carrillo de Santa Pau

Jun 2019

NGSCAT2. AN IMPROVED TOOL TO ASSESS THE EFFICIENCY OF TARGETED ENRICHMENT SEQUENCING

Autor: Alejandro García Sánchez
Tutor: Javier Pérez Florido
Ponente: Enrique Carrillo de Santa Pau

Clinical Bioinformatics Area, Junta de Andalucía
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Jun 2019

Resumen

Resumen

Desde la secuenciación del primer genoma humano, el desarrollo de nuevas técnicas de secuenciación ha crecido exponencialmente, bajando los costes de esta por debajo de los mil dólares. Actualmente, en el ámbito del diagnóstico clínico se utilizan tecnologías de segunda generación para secuenciar regiones determinadas del genoma, ya sea la totalidad de los genes (exoma) o un conjunto de ellos (panel de genes). La secuenciación de regiones concretas del genoma se conoce como secuenciación masiva dirigida. Esta técnica incluye, entre otras, una etapa de captura y enriquecimiento de las regiones a secuenciar, cuya eficiencia condicionará el resultado final de la secuenciación. Por ello, son necesarias herramientas de software que evalúen la eficiencia y la ausencia de sesgo en el proceso de enriquecimiento, actuando como control de calidad antes de continuar con el análisis de los datos obtenidos. Hoy en día existen varias herramientas que permiten realizar este control de calidad, entre las que destaca *ngsCAT* (Next Generation Sequencing data Capture Assessment Tool). A pesar de ser una herramienta muy extendida en la comunidad científica), presenta limitaciones en cuanto a su arquitectura, tecnología usada, instalación y representación de los datos. Por tanto, en este trabajo se ha desarrollado una versión llamada *ngsCAT2*, la cual suple las carencias que presentaba su predecesora. Esto se ha conseguido gracias a una nueva arquitectura, al uso de *Python3*, el empleo de gráficas interactivas, la adición de nuevas funcionalidades y la publicación en plataformas colaborativas. Con ello, se ha conseguido una herramienta más completa, flexible y sencilla de desarrollar, facilitando su utilización por parte de usuarios inexpertos que deseen evaluar el proceso de captura de sus experimentos de secuenciación dirigida, y a desarrolladores que deseen ampliar la herramienta. Todas estas nuevas características permiten un aumento del soporte del software, ampliando su uso y permanencia en la comunidad científica.

Palabras Clave

Secuenciación Masiva Dirigida, Exoma, Panel de genes, Python, Herramienta control calidad

Abstract

Since the first sequencing of human genome, the development of new sequencing techniques has grown exponentially, lowering costs below a thousand American Dollars. Currently, second-generation technologies are used in the field of clinical diagnosis to sequence specific regions of the genome, allowing us to sequence either the entirety of genes (exoma) or a particular set of them (gene panel). Sequencing specific regions of the genome is known as targeted mass sequencing, which includes a capturing and enriching step of the regions of interest within the genome. The efficiency of this enrichment process will condition the sequencings final results. Therefore, software tools are needed to evaluate the efficiency and lack of bias during the enrichment process, acting as quality control before continuing with the analysis of the data obtained. Nowadays there are several tools that allow us to perform this quality control, such as *ngsCAT* (Next Generation Sequencing data Capture Assessment Tool). Despite its use being widespread among the scientific community, this tool has limitations due to its architecture, the technology used, installation and data representation. Because of this, we have developed a new version called *ngsCAT2*, which solves the limitations showed by its predecessor. This has been achieved thanks to its new architecture, the use of *Python3* and interactive graphics, the addition of new functionalities and its publication on collaborative platforms. This has resulted in a more complete, flexible and easy to develop tool, facilitating its use by inexperienced users who wish to evaluate the capture process of their sequencing experiments, and to developers who would like to extend the tool. All these new features allow for an increased support of the tool, extending its use and permanence among the scientific community.

Key words

Targeted Next-generation Sequencing, Exome, Panel of genes, Python, Control Quality Tool

Índice general

Índice de Figuras

Índice de Tablas

1

Introducción

1.1. Motivación del proyecto

Desde la finalización del proyecto genoma humano en el año 2003, se ha producido un enorme progreso en las tecnologías de secuenciación masiva (Next Generation Sequencing, NGS), de forma que se han realizado avances importantes en el estudio del contenido, organización, función y evolución de la información genética en un genoma completo. A la par, el coste de la secuenciación se ha ido reduciendo de forma significativa (fig.1.1), siendo posible en la actualidad secuenciar un genoma completo por aproximadamente 1000 dólares americanos (1).

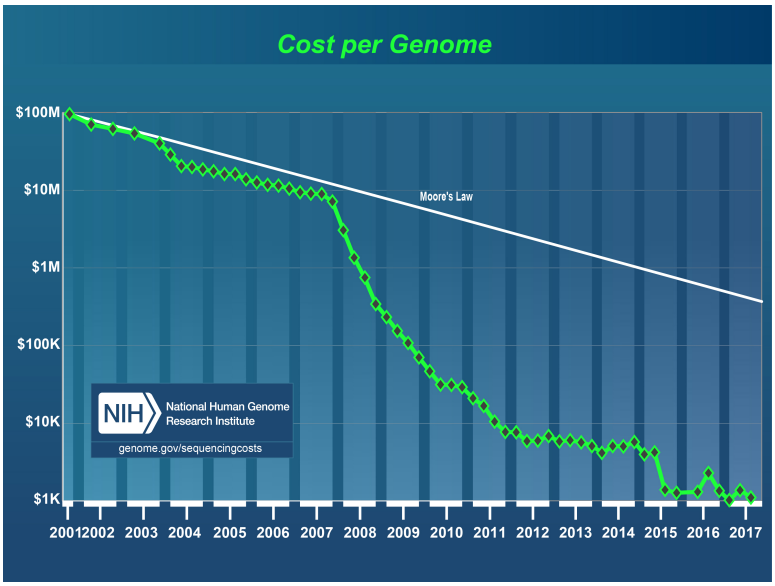

Figura 1.1: Precio de la secuenciación de genoma completo a lo largo de los años, linea verde. Representado con la linea blanca podemos observar el precio esperado según la ley de Moore. Imagen obtenida de (1)

En el ámbito del diagnóstico genético es común secuenciar mediante NGS regiones específicas del genoma(2; 3; 4). Estas regiones pueden corresponder a regiones codificantes del genoma

(conocido como exoma), a un conjunto de genes asociados a numerosas enfermedades (exoma clínico)(5) o simplemente a un número más o menos variable de genes de interés (panel de genes)(6; 3). A esta secuenciación de regiones específicas se le conoce comúnmente como **secuenciación masiva dirigida** y es una herramienta útil para interrogar en un solo experimento varios loci o incluso todas las regiones genómicas codificantes a un coste relativamente bajo.

El enriquecimiento de regiones para la secuenciación masiva dirigida incluye una etapa de captura donde las sondas se unen a las regiones de interés. El éxito de la secuenciación dependerá estrechamente de la eficiencia de esta captura y el enriquecimiento de dichas regiones. El grado de enriquecimiento puede ser calculado antes de la secuenciación mediante una reacción en cadena de polimerasa (PCR, Polymerase Chain Reaction) cuantitativa para un número reducido de regiones de interés. Únicamente el análisis de los datos obtenidos mediante la secuenciación masiva nos aportará el grado de enriquecimiento para todas las regiones de interés (Regions of Interest, ROIs), proporcionando información de todo el proceso de captura y la existencia de posibles sesgos.

Existen diversidad de estudios que establecen cuáles son los aspectos críticos para evaluar el proceso de secuenciación dirigida(7; 8; 9): **(i) sensibilidad**, que permite conocer cómo de bien están cubiertas las regiones de interés, **(ii) especificidad**, para medir la proporción de las secuencias o *reads* que se encuentran fuera de las regiones de interés y **(iii) uniformidad**, para detectar la existencia de sesgos en las regiones de interés. En la literatura, existen varios programas de software que permiten la evaluación del proceso de enriquecimiento en experimentos de secuenciación dirigida (véase sección 2.2) en términos de sensibilidad, especificidad y/o uniformidad, destacando *ngsCAT* (Next Generation Assessment Tool)(10). A pesar de ser una herramienta usada por la comunidad científica(11), necesita mejorar ciertos aspectos como una mayor facilidad de instalación, la interacción con el usuario, más funcionalidad y la posibilidad de permitir un desarrollo colaborativo.

1.2. Objetivos

En este sentido, el objetivo de este trabajo es el desarrollo de una versión mejorada de *ngsCAT* que cumpla con los puntos indicados anteriormente, más concretamente:

- Migración del código fuente de *Python* 2 a *Python* 3.
- Agregar soporte para las herramientas estándares de gestión de paquetes de *Python* (pip) para la instalación del ejecutable y sus dependencias.
- Agregar figuras interactivas por medio de paquetes gráficos actuales como *Plotly Python Library* (12).
- Incluir nuevas métricas relacionadas con la evaluación del proceso de enriquecimiento.
- Incluir el código fuente de la herramienta en un sistema de control de versiones (Git) y alojar dicho código en GitHub.
- Adaptar la herramienta y sus mejoras para su ejecución en ordenadores personales con baja potencia de cómputo.

Para la ejecución de estos objetivos ha sido necesario realizar una refactorización del código enfocada a la mejora del alto acoplamiento y la estructura poco modularizada.

Con estas mejoras, se pretende que la herramienta pueda ser utilizada por profesionales no expertos en bioinformática que trabajen con experimentos de secuenciación dirigida. Además, estas mejoras permitirán el continuo desarrollo y mejora de la herramienta de un modo colaborativo, ampliando así su soporte y penetrancia en la comunidad científica.

1.3. Estructura de la memoria

Este trabajo está estructurado de la siguiente forma: para comenzar en el capítulo 2 se introducirá conceptos de secuenciación masiva así como los diferentes programas informáticos destinados a la evaluación del proceso de enriquecimiento en experimento de secuenciación masiva dirigida. Posteriormente, se realizará una análisis completo de la herramienta *ngsCAT*. En el capítulo 3 se describirá el desarrollo de la nueva versión de la herramienta, *ngsCAT2*, en términos de arquitectura, y las nuevas funcionalidades. Finalmente, en el capítulo 4 veremos el resultado de la ejecución completa de la nueva herramienta tomando como ejemplo un exoma Humano. Además, se realizará una prueba de rendimiento comparativa con su predecesora.

2

Secuenciación masiva dirigida, Herramientas software para la evaluación del proceso de enriquecimiento

2.1. Introducción a la secuenciación de DNA

La secuenciación de las moléculas de Acido desoxiribonucleico (DNA) es el proceso de determinar el orden preciso de los nucleótidos de una molécula de DNA. Engloba a cualquier metodología o técnica capaz de determinar el orden de los desoxiribonucleótidos de Adenina, Citosina, Guanina y Timina en una de las hebras del DNA (la cadena complementaria se puede conocer utilizando el principio de complementariedad de bases).

El conocimiento de la secuencia de DNA es indispensable para la investigación biológica y para numerosas disciplinas como genómica funcional, transcriptómica, oncología, microbiología y biología forense, entre otras. Por esta razón, desde el descubrimiento de la molécula de DNA en 1953 se han realizado grandes esfuerzos para desarrollar tecnologías que puedan determinar el orden exacto de las bases nucleotídicas en una molécula de DNA. Podemos dividir la metodologías o tecnologías de secuenciación en tres etapas o generaciones: metodologías de primera, segunda y de tercera generación.

Las metodologías de secuenciación de primera generación marcaron el inicio de la era. En 1977, Maxam y Gilbert (13) desarrollaron la secuenciación por rotura o el también llamado "método por degradación química". Esta metodología no se ha usado de forma muy amplia debido a la elevada toxicidad de algunos reactivos químicos utilizados y la necesidad de disponer de altas cantidades de DNA. El otro método que se engloba en las técnicas de primera generación es la secuenciación "dideoxi" o por terminación de cadena, desarrollada por el dos veces ganador del premio Nobel de química Frederick Sanger en 1975 pero comercializada posteriormente en 1977. El método de *Sanger* (14) es un método enzimático que permite determinar la secuencia de una hebra de DNA (llamada molde) a medida que se va sintetizando su cadena complementaria y, al igual que el método de *Maxam-Gilbert*, se requiere del uso de electroforesis en gel para poder obtener el orden de los nucleótidos que componen el fragmento a secuenciar.

Gracias a la paralelización y automatización de la metodología en la que se basa la secuenciación Sanger, se pudo completar la secuenciación del genoma humano en 2003. El "proyecto

genoma humano" requirió de 13 años y una gran cantidad de recursos humanos y tecnológicos para completarse (15). La ingente cantidad de recursos requeridos demandó la necesidad de nuevas tecnologías de secuenciación que permitieran la generación masiva de secuencias, una reducción del tiempo de procesado y una reducción del coste de todo el proceso. Así, gracias a un programa de subvenciones del *National Human Genome Institute* (NHGRI) de Estados Unidos, a partir del año 2005 se empezaron a comercializar los primeros secuenciadores que generaban secuencias de DNA de forma masiva a un coste significativamente menor. Estos secuenciadores se conocen como tecnologías de segunda generación o *Next Generation Sequencing* (16).

Las tecnologías de segunda generación comparten tres aspectos y mejoras con respecto a las metodologías de primera generación:

1. Requieren de un pre-procesado del DNA o creación de una librería que permite preparar el DNA para su carga en el secuenciador.

2. Se producen de miles a millones de reacciones de secuenciación en paralelo en un proceso basado en ciclos.

3. La salida del proceso de secuenciación se realiza de forma directa por lo que no es necesario la utilización de electroforesis en gel.

Las tecnologías que componen la segunda generación se basan en amplificaciones de los fragmentos de DNA que componen la librería mediante la técnica de reacción en cadena de polimerasa (*Polymerase Chain Reaction*, PCR). De esta forma, se pueden detectar las señales emitidas en las reacciones químicas que ocurren en el proceso de secuenciación. Este grupo está conformado por las siguientes tecnologias: 454 (pirosecuenciación) de *Roche Diagnostics* y SOLiD (*Sequencing by Oligonucleotide Ligation and Detection*) de *Life Technologies*, ambas actualmente fuera de mercado. También encontramos enmarcadas en esta categoría a los secuenciadores *(*Ion Torrent/Proton) de *ThermoFisher* y a los secuenciadores de *Illumina* (17). De todas estas tecnologías, los secuenciadores de *Illumina* son, con diferencia, los más utilizados por la comunidad científica, por su alta precisión, coste de secuenciación por base, aplicaciones y versatilidad. Tal es su potencial que gracias a un conjunto de secuenciadores del modelo de *Hiseq X* se ha conseguido disminuir el tiempo de secuenciación del genoma humano completo hasta las 72 horas con un coste cercano a los 1000 dólares americanos. Illumina pretende mediante su nuevo modelo de secuenciador *NovaSeq6000* (fig. 2.1) bajar los tiempos de secuenciación hasta las 60 horas en el caso de genoma completo.

Figura 2.1: Secuenciador *NovaSeq 6000*

Finalmente, existen otro tipo de tecnologías dentro de la secuenciación masiva conocidas como de tercera generación. Estos métodos comparten dos características principales:

1. No están basadas en la amplificación de los fragmentos mediante PCR para la secuenciación.

2. El proceso de secuenciación se produce en tiempo real, ya que no se basa en ciclos.

Este grupo lo componen principalmente dos tecnologías, SMRT (*single molecule real time*) de *Pacific Biosciences* y un sistema de secuenciación basado en nanoporos desarrollado por *Oxford nanopore*.

SMRT (18) es un método de secuenciación en tiempo real de una única molécula de DNA. Con esta tecnología, la preparación de las librería es rápida (en torno a 6 horas) y la principal ventaja que tiene es una longitud de secuencias o *reads* mucho mayor que las obtenidas con los métodos basados en amplificación por PCR. La principal desventaja es un menor número de *reads* por ejecución o *througtput* en comparación con los secuenciadores de lecturas cortas como *Illumina* y la alta tasa de errores, alrededor del 14 % con el modelo *PacBio RS II*, aunque con el modelo de secuenciador más actual, *Sequel*, se está consiguiendo reducir. Por otro lado, tenemos la tecnología basada en Nanoporos desarrollada por *Oxford Nanopore*, en la cual la cadena de DNA es introducida a través de un poro en una membrana de material conductor. Cada base o nucleótido al pasar por el nanoporo genera un cambio en la conductividad del material que puede ser detectado. En 2014, *Oxford nanopore* sacó al mercado un dispositivo portátil basado en la tecnología de nanoporos llamado *minION* y están desarrollando plataformas para conectar y paralelizar varios de estos secuenciadores.

Como hemos visto, las tecnologías de secuenciación masiva han evolucionando exponencialmente (fig. 2.2). Se ha incrementado la longitud de *reads* llegando a obtener subconjuntos de lecturas de hasta 60 Kilo bases con*PaCBIO RSII*. En cuanto a cantidad de datos generados (*throughtput*) *Illumina* mediante el secuenciador *HiSeq X Ten* es capaz de generar 1,8Tb (*terabase*) por ejecución de secuenciador y precisamente con este modelo se ha conseguido reducir el coste de la secuenciación de una manera sustancial.

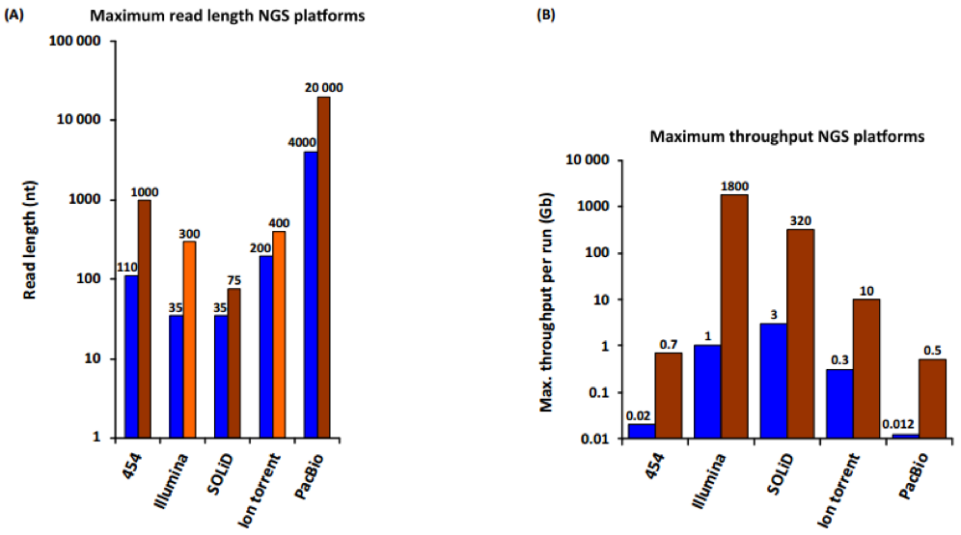

Figura 2.2: Evolución de las plataformas de NGS. (A) Las barras azules representan la longitud de *reads* máxima en el primer secuenciador comercializado por 454, Illumina/Solexa, SOLiD (Life Technologies), Ion Torrent (Life Technologies) y Pacific Biosciences. Las barras naranjas representan la longitud de reads máxima que pueden ser obtenidas con esas tecnologías en el año 2014. (B) Máximo throughput del primer instrumento de secuenciación disponible comercialmente (barra azul) y del instrumento comercializado en 2014 (barra naranja) Figura obtenida y modificada de (17).

2.1.1. Aplicaciones de las tecnologías de secuenciación masiva

Es indiscutible el potencial de las tecnologías de NGS en disciplinas como la biotecnología, la biología o la biomedicina. La escalabilidad, los precios relativamente asequibles y la eficiencia de la secuenciación masiva están proporcionando un progreso sin precedentes en, por ejemplo, el diagnóstico de enfermedades raras o cáncer o en la predicción de un efecto de un fármaco sobre el paciente (19; 20).

Las aplicaciones de las tecnologías de NGS son muy variadas. Por ejemplo, la secuenciación de las moléculas de ácido ribonucleico (RNA), también conocido como RNA-seq, nos permite obtener la secuencia completa de los RNAs presentes dentro de la muestra. Mediante esta técnica podemos conocer los niveles de expresión génica, o el número de isoformas de un gen presentes de una muestra. Además de los RNA mensajeros obtenemos la información de la secuencia de RNA ribosómicos, RNA de transferencia o RNA no codificantes. Se puede también estudiar los sitios de unión proteína-DNA mediante ChiP-seq (inmunoprecipitación de la cromatina), permitiendo estudiar la compleja regulación génica. Otra de las aplicaciones es la detección cambios químicos en las bases nucleotídicas como la metilación. También, mediante la NGS, podemos caracterizar el genoma de una especie en particular cuyo genoma de referencia no es conocido. A esta aplicación se le conoce como *ensamblado de novo de genoma completo*.

Sin embargo, nos vamos a centrar en la secuenciación de genomas ya conocidos, distinguiendo entre:

1. Secuenciación de genoma completo o WGS (*whole genome sequencing*): donde se re-secuencia el genoma completo del organismo bajo estudio. Debido a la reducción de costes, en la actualidad es abordable la secuenciación de genomas completos a nivel poblacional. En este sentido, destacamos iniciativas como el proyecto 1000 Genomas de Navarra (21) o el proyecto de los 100.000 genomas de Reino Unido (22), entre otros.

2. Secuenciación dirigida: En algunos casos ya sea por una cuestión económica o por el objetivo del experimento, se puede recurrir a la re-secuenciación de determinadas regiones de

interés. En este caso, las regiones de interés son selectivamente capturadas y enriquecidas (mediante hibridación a la cadena de DNA complementaria) antes de la secuenciación de forma que se obtienen datos de secuenciación únicamente de las regiones capturadas (20; 5). Dentro de la secuenciación masiva dirigida podemos realizar la re-secuenciación del exoma completo o WES (*Whole Exome Sequencing*) donde únicamente son capturadas y secuenciadas todas las regiones genómicas que codifican proteínas. Estas regiones son conocidas como exones y el genoma humano completo aproximadamente contiene 180.000 de estas regiones o 30 Megabases. Al conjunto de exones se le denomina exoma que representa menos de un 2 % del genoma humano y aunque puede parecer una cifra baja, aproximadamente un 85 % de las mutaciones conocidas causantes de enfermedades mendelianas se encuentran en las regiones codificantes (23). Además de la secuenciación del exoma, a través de la secuenciación dirigida podemos secuenciar únicamente un determinado conjunto de genes de interés, conocido comúnmente como "panel de genes". Los paneles de genes normalmente engloban a decenas o cientos de genes relacionados con una o más patologías a estudiar. En el ámbito hospitalario estos paneles son ampliamente utilizados para el diagnóstico y estudio de enfermedades como el Cáncer de mama (3).

A priori, la gran cantidad de información generada por la secuenciación de genoma completo puede ser una ventaja, pero el coste de la propia secuenciación y la gran cantidad de datos que posteriormente tienen que ser analizados, consumen una gran cantidad de recursos económicos y humanos (24). Por estos motivos y porque en muchas ocasiones el objetivo del experimento no requiere la re-secuenciación del genoma completo, se plantea como alternativa la secuenciación dirigida de regiones de interés. Por ejemplo, es común utilizar esta técnica en el diagnóstico genético, secuenciando determinados genes de interés o exomas completos.

Por otro lado, el enriquecimiento de regiones para la secuenciación masiva dirigida incluye una etapa de captura donde las sondas se unen a las regiones de interés y complementarias de DNA genómico. El éxito de la secuenciación dependerá estrechamente de la eficiencia de esta captura y el enriquecimiento de dichas regiones. El grado de enriquecimiento puede ser calculado antes de la secuenciación mediante una reacción en cadena de polimerasa cuantitativa (qPCR) para un número reducido de regiones de interés. Únicamente el análisis de los datos obtenidos mediante la secuenciación masiva nos aportará el grado de enriquecimiento para todas las regiones de interés (ROIs), proporcionando información de todo el proceso de captura y la existencia de posibles sesgos.

Existen diversidad de estudios que establecen cuáles son los aspectos críticos para evaluar el proceso de secuenciación dirigida (7; 8; 9): **(i) sensibilidad**, que permite conocer cómo de bien están cubiertas las regiones de interés, **(ii) especificidad**, para medir la proporción de las secuencias o *reads* que se encuentran fuera de las regiones de interés y **(iii) uniformidad**, para detectar la existencia de sesgos en las regiones de interés.

2.2. Programas para la evaluación de la eficiencia del proceso de enriquecimiento.

En la comunidad científica existen varias herramientas software que permiten la evaluación del proceso de enriquecimiento en experimentos de secuenciación dirigida en términos de sensibilidad, especificidad y/o uniformidad. Algunas herramientas de manejo de secuencias contienen funcionalidades útiles para obtener métricas determinadas, como por ejemplo *BedTools* (25) o *PicardTools* (26) que pueden aportar estadísticas generales de la cobertura o del sesgo del contenido GC. Estas herramientas de manejo de secuencias normalmente no aportan todas las métricas necesarias para un realizar un control de calidad exhaustivo, obligando al investigador a crear programas complejos que agrupen las métricas de los diferentes programas.

Por otro lado, existen otras herramientas más completas para la evaluación del proceso de captura aportando métricas específicas para experimentos de secuenciación dirigida. Algunas de estas herramientas son: *NGSrich* (27), TarSeqQC (28), TEQC (29), Coverview (30) y *ngsCAT* (10) de las cuales hablaremos a continuación.

NGSRich es un programa desarrollado en *Java* y diseñado para integrarse en entornos de supercomputación. Los resultados se generan en un archivo HTML facilitando así su visualización, además aporta datos como la calidad de la secuenciación. Sin embargo, *NGSRich* no ha recibido actualizaciones recientes y no se encuentra disponible en plataformas de desarrollo colaborativo como GitHub.

TarSeqQC es un paquete de *R* que permite realizar un control de calidad y una rápida exploración de los resultados de experimentos de secuenciación dirigida en términos de cobertura. Aunque se trata de una herramienta con más de 1500 descargas en 2018 `http://bioconductor.org/packages/stats/bioc/TarSeqQC/`, está diseñada para explorar un pequeño grupo de regiones específicas y aunque es suficiente para realizar análisis de paneles de genes, dificulta su uso para el análisis de exomas completos. Este paquete se encuentra disponible en el repositorio *Bioconductor*, que es el repositorio de paquetes de R donde se alojan más herramientas destinadas al análisis de datos biológicos.

TECQC, al igual que la herramienta anterior, es otro paquete de *R* disponible en el repositorio Bioconductor. El principal problema de este paquete es su alto consumo de memoria RAM lo cual no lo hace apto para el análisis de todas las regiones de un exoma (50Mb).

Coverview es una herramienta en la que a través de su interfaz interactiva, el usuario puede visualizar una gran cantidad de información relacionada con la cobertura o mediciones de calidad de la secuenciación a nivel gen e incluso a nivel de base nucleotídica. Está escrita en *python2* y utiliza el *framework Flask* (31) para generar la interfaz, requiriendo el uso de un servidor local. La principal limitación de esta herramienta viene dada por la falta de figuras que aporten una visión global de la evaluación del proceso de enriquecimiento. (30).

Entre las herramientas disponibles para la evaluación del enriquecimiento en la secuenciación dirigida destaca *ngsCAT* (next generation sequencing data Capture Assessment Tool). Esta herramienta proporciona una gran diversidad de métricas y permite su ejecución en ordenadores con recursos limitados. En la siguiente sección 2.3 hablaremos en profundidad de sus diferentes funciones y los mecanismos que utiliza para llevarlas a cabo.

2.3. ngsCAT

ngsCAT es una aplicación de línea de comandos escrita en *Python* que facilita una evaluación del enriquecimiento en la secuenciación dirigida en términos de:

- Sensibilidad, que nos permiten conocer cómo de bien están cubiertas las regiones de interés.
- Especificidad, para medir la proporción de secuencias o *reads* alineadas en regiones que no corresponden a las de interés.
- Uniformidad, para detectar posibles sesgos en las regiones de interés.

ngsCAT únicamente requiere de un archivo de alineamientos en formato binario que represente los alineamientos de las secuencias con respecto a un genoma de referencia *BAM (Binary alignment map)* y un archivo *BED (Browser extensible data)* donde se describen las regiones de interés, que representan las zonas capturadas y posteriormente secuenciadas. La herramienta puede manejar datos de la mayoría de plataformas de secuenciación y puede ser fácilmente integrable en flujos de análisis de datos o *pipelines*. Además *ngsCAT* cuenta con la paralelización de sus tareas lo cual permite que se realice todo el análisis haciendo uso de multihilo. *ngsCAT* genera como salida un informe en formato HTML dividido en las siguientes secciones (sensibilidad, especificidad y uniformidad). En cada una de las secciones se presentan una serie de tablas y figuras, así como los diferentes parámetros utilizados. Por otro lado, *ngsCAT* genera al usuario diversos archivos en formato *bedGraph* con información de la profundidad en las regiones de los diferentes cromosomas o *contigs*. Estos archivos pueden ser cargados en visualizadores genómicos como *Integrative Genomics Viewer* (*IGV*) o el proporcionado por *UCSC*.

2.3.1. Funcionalidad

A continuación haremos un breve repaso a cada una de las métricas que utiliza el programa *ngsCAT* para la evaluación del enriquecimiento en la secuenciación dirigida en términos de sensibilidad, especificidad y uniformidad.

Sensibilidad

ngsCAT aporta dos métricas que evalúan la calidad de la cobertura en las regiones de interés. La primera de ellas es el porcentaje de bases cubiertas a diferentes umbrales de profundidad que nos permite evaluar la sensibilidad del proceso de captura estudiando el grado de regiones cubiertas a una profundidad determinada. Como podemos ver en la figura 2.3

La segunda métrica dentro de la sección de Sensibilidad se trata de una métrica opcional y proporciona una medida de saturación, que permite conocer si aumentando la profundidad de la secuenciación se obtendría una mejora en los resultados en cuanto al porcentaje de posiciones cubiertas. La gráfica generada por esta métrica la podemos ver en la Figura 2.4.

Especificidad

En términos de especificidad, *ngsCAT* nos permite conocer cuanto esfuerzo de secuenciación es empleado en capturar zonas fuera de las regiones de interés. Estos datos son representados mediante un gráfico de barras (Fig. 2.5). Además proporciona el número de *reads* duplicados (lecturas que alinean exactamente en la misma región del genoma de referencia) dentro *on-target* como fuera de las regiones de interés *off-target* (Figura 2.6). Por otro lado *ngsCAT*

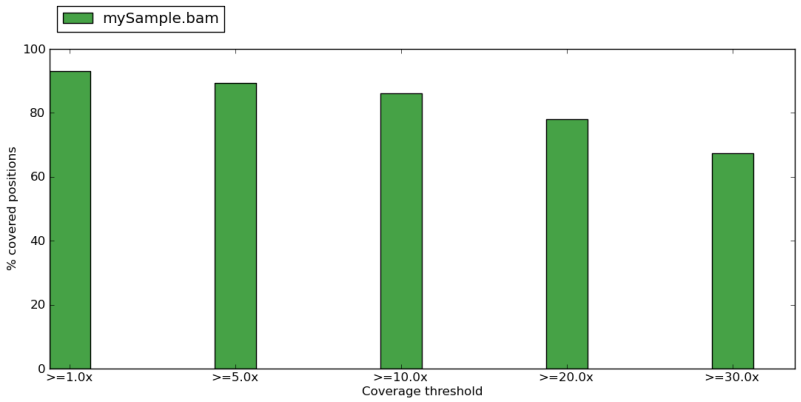

Figura 2.3: Porcentaje de bases cubiertas a diferentes umbrales de profundidad. En el eje Y está representada la proporción de todas las bases que conforman las regiones de interés. En eje X están representados los diferentes umbrales de profundidad. En este caso más de un 90 % por ciento de todos los nucleótidos que forman nuestras regiones de interés están cubiertos por una o más lecturas o *reads*.

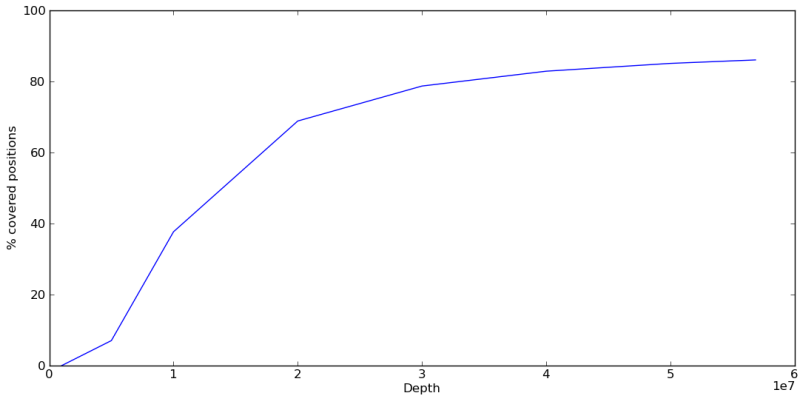

Figura 2.4: Gráfica de saturación, en el eje x están representado el número de *reads* o secuencias del experimentos de secuenciación y en el eje y el porcentaje de bases cubiertas a más de 10x de profundidad

proporciona la información regiones con alta profundidad fuera de las regiones de interés (a más de 1 Kilobase y por defecto con más de 15x de profundidad), mediante los archivos generados por la herramienta en formato BED (extension ".off.bed").

Uniformidad

ngsCAT proporciona cuatro métricas que permiten conocer si las lecturas están distribuidas uniformemente en las regiones de interés, permitiéndonos saber si hay sesgo debido a localizaciones genómicas o composiciones nucleotídicas. Comenzaremos por la distribución de profundidad por posición, esta métrica proporciona dos gráficas. La primera de ellas nos muestra la distribución de la profundidad por base capturada para solo aquellas bases con una profundidad de al menos 1x (Figura 2.7). La segunda gráfica es una representación de la distribución de la profundidad por base en forma de gráfico de cajas o boxplot (Figura 2.8).

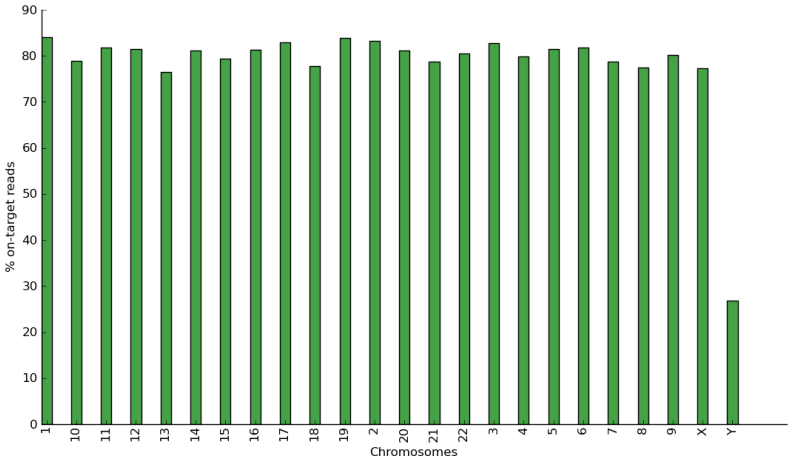

Figura 2.5: Gráfica de reads que cubren las regiones de interés,donde están representadas las lecturas alineadas en el genoma de referencia y dentro de las regiones de interés en cada uno de los cromosomas

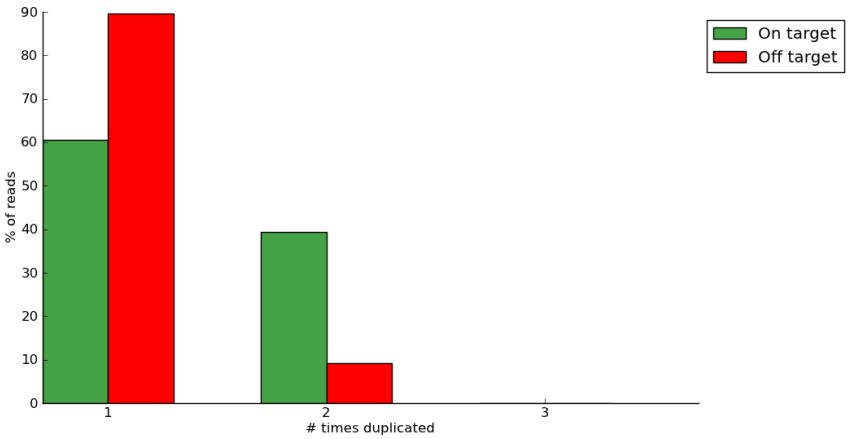

Figura 2.6: Gráfica de multiplicidad de las lecturas. En el eje x está representado el número de veces de aparición de lecturas que alinean de la misma forma en el genoma de referencia. Las barras de color verde y rojo representan a las lecturas que mapean dentro *on-target* y fuera de las regiones *off-target* de interés respectivamente.

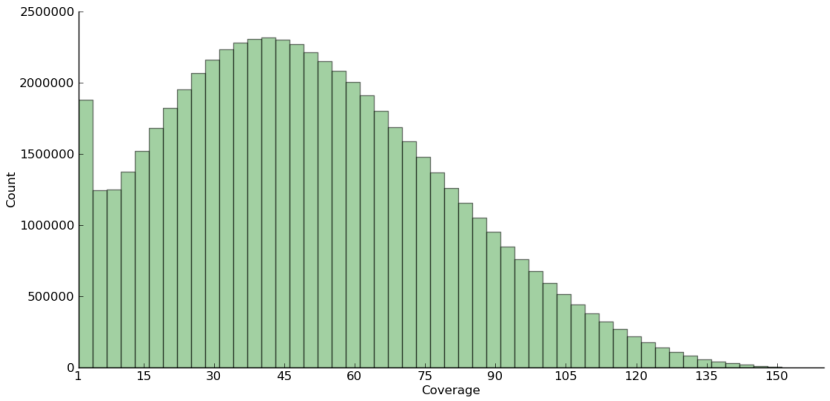

Figura 2.7: Histograma de distribución de la profundidad por posición o base.

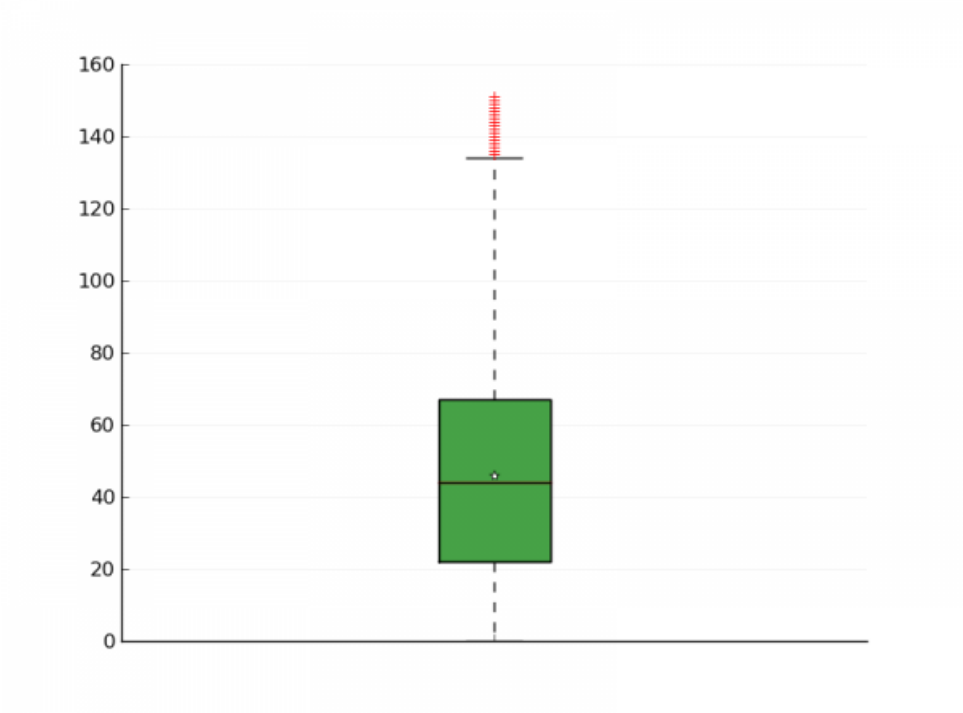

Figura 2.8: Boxplot de la distribución de la profundidad por posición o base.

Por otro lado en términos de uniformidad *ngsCAT* nos proporciona la distribución de la profundidad en las regiones de interés por cada cromosoma como podemos ver en la figura 2.9

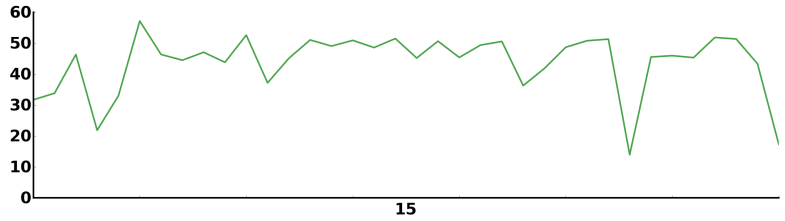

Figura 2.9: Gráfica de distribución de la profundidad a lo largo del las regiones de interés en el cromosoma

Otra de las métricas relacionadas con la uniformidad es la distribución de las desviaciones estándar de las profundidades en las regiones de interés. En esta métrica para cada una de las regiones son calculadas la media y la desviación estándar de la profundidad de cada una de las bases que la componen proporcionando un valor de desviación estándar normalizado (desviación estándar / media). Todas estas regiones normalizadas son representadas en un histograma y un boxplot (figura 2.10).

Otra métrica opcional de *ngsCAT* permite conocer la distribución de las profundidades en función del contenido en GC. En ella son calculadas las medias de la profundidad de las regiones y el porcentaje de contenido en GC presente en ellas. Esta gráfica nos permite medir la uniformidad en términos de la composición nucleotídica (Figura 2.11) Finalmente, en el caso de realizar una evaluación de dos experimentos a la vez se calcula y representa en forma de gráfica la correlación de profundidad por región de interés 2.12, entre ambos archivos BAM.

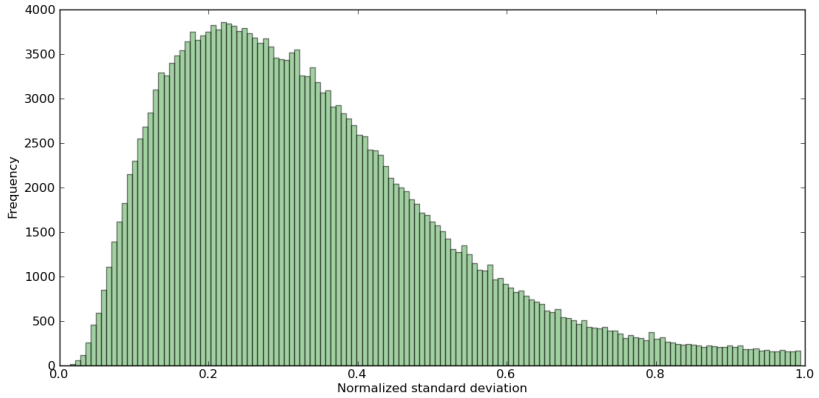

Figura 2.10: Distribución de las desviaciones estándar de las profundidades de las regiones de interés

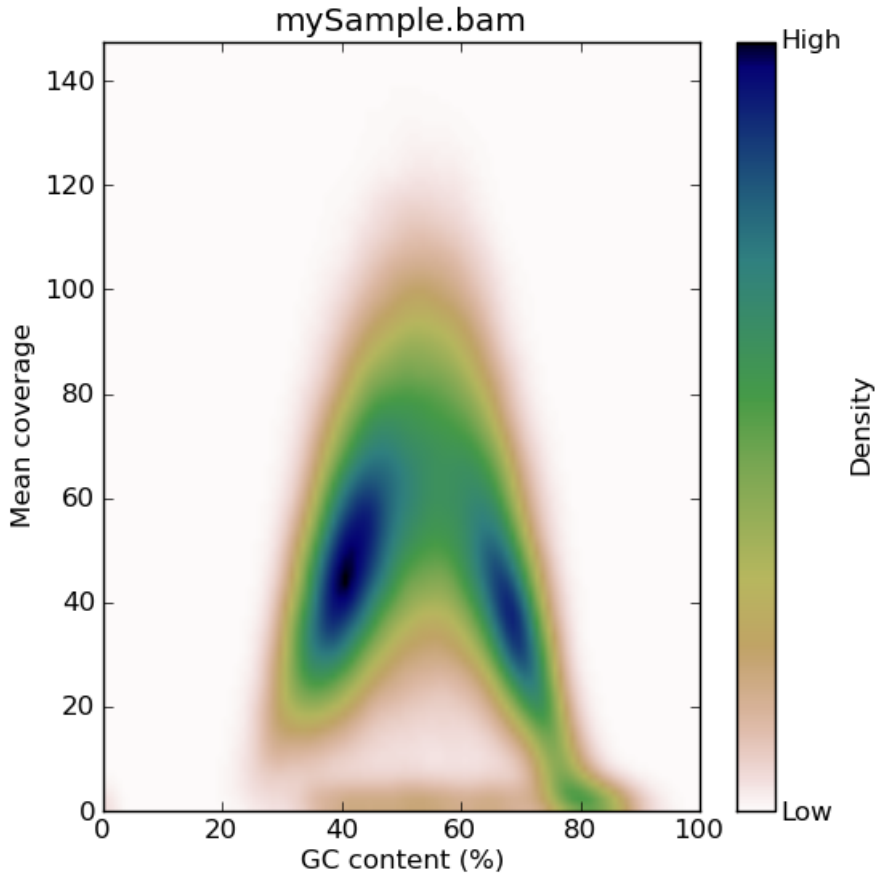

Figura 2.11: Gráfica de contenido en GC. En ella están representadas la distribución de las medias de profundidad de las regiones frente al contenido en GC que tienen cada una de ellas

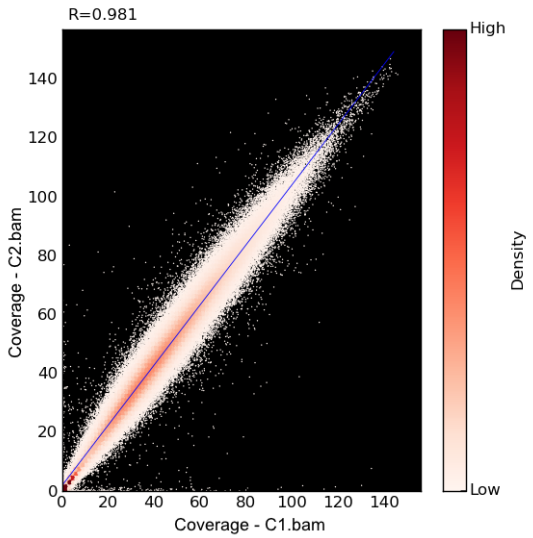

Figura 2.12: Gráfica de correlación de profundidad por Región de Interés.

Una vez comentada cuál es la funcionalidad de *ngsCAT* en las siguientes secciones vamos a describir desde un punto de vista general aspectos como la arquitectura, flujo y control de concurrencia, comunicación entre procesos y organización a nivel de directorios de la herramienta.

2.3.2. Análisis de la herramienta

Dependencias

ngsCAT requiere un intérprete de *Python* 2 (ha sido testado en las versiones 2.6 y 2.7) y de librerías ampliamente utilizadas en el área de la bioinformática tales como:

- *Samtools* es una *suite* de programas para el manejo de datos de secuenciación. En concreto permite leer, escribir, editar, indexar y visualizar los formatos que contienen información de alineamientos SAM/BAM/CRAM. (32)
- *pysam* es un módulo *Python* que permite leer y manipular archivos SAM/BAM cuenta con una gran cantidad de usuarios debido a que engloba a la herramienta *Samtools*.
- *Numpy, Scipy* paquete fundamental para computación científica con *Python* y paquete que provee de rutinas para integración numérica, estadística...
- *Matplotlib* forma parte del mismo ecosistema de las dependencias anteriores y permite la creación de figuras de alta calidad.
- *xlwt* para la generación de archivos en formato xls.

Flujo de ejecución y estructura de ngsCAT

Para comprender la estructura de *ngsCAT*, primero, es necesario comprender el flujo de trabajo de esta. *ngsCAT* parte de uno o dos ficheros de alineamiento de secuencia binario en formato BAM (*Binary Alignment/Map*) (32) que contienen el resultado del alineamiento de las secuencias o *reads* frente a un genoma de referencia y un archivo con formato BED (Browser Extensible Data) https://www.ensembl.org/info/website/upload/bed.html que contendrá

la localización en el genoma de las regiones de interés (regiones capturadas). El punto de entrada del programa o *main* se sitúa en *ngscat.py*. Para comenzar, la herramienta comprueba si el formato de los ficheros de entrada es el correcto y en caso positivo comienza con el análisis. Como se puede ver en la figura 2.13 la primera parte de los cálculos es la generación de los datos de la profundidad en las regiones de interés. Para ello, el archivo BAM es recorrido lectura por lectura y calcula la cobertura y profundidad de las regiones de interés base a base nucleotídica. Estos resultados de profundidad se van escribiendo en un archivo de texto plano con un formato determinado. Posteriormente se van ejecutando de una manera concurrente los diferentes módulos que calcularán las diferentes métricas en términos de sensibilidad, especificidad y uniformidad. Cada una de los módulos métrica-informe toman los datos de profundidad anteriormente calculados y generan resultados de diferente tipo como: variables que contienen información para el informe final, archivos de regiones (ficheros en formato BED), tablas-resumen (archivos en XLS) y las diferentes gráficas (archivos PNG). Finalmente, una vez que han finalizado todos los procesos correspondientes a los diferentes módulos, una plantilla HTML es utilizada para plasmar todos los resultados, generando por tanto un informe final que muestra de forma sencilla la evaluación del proceso de enriquecimiento (Figura 2.13)

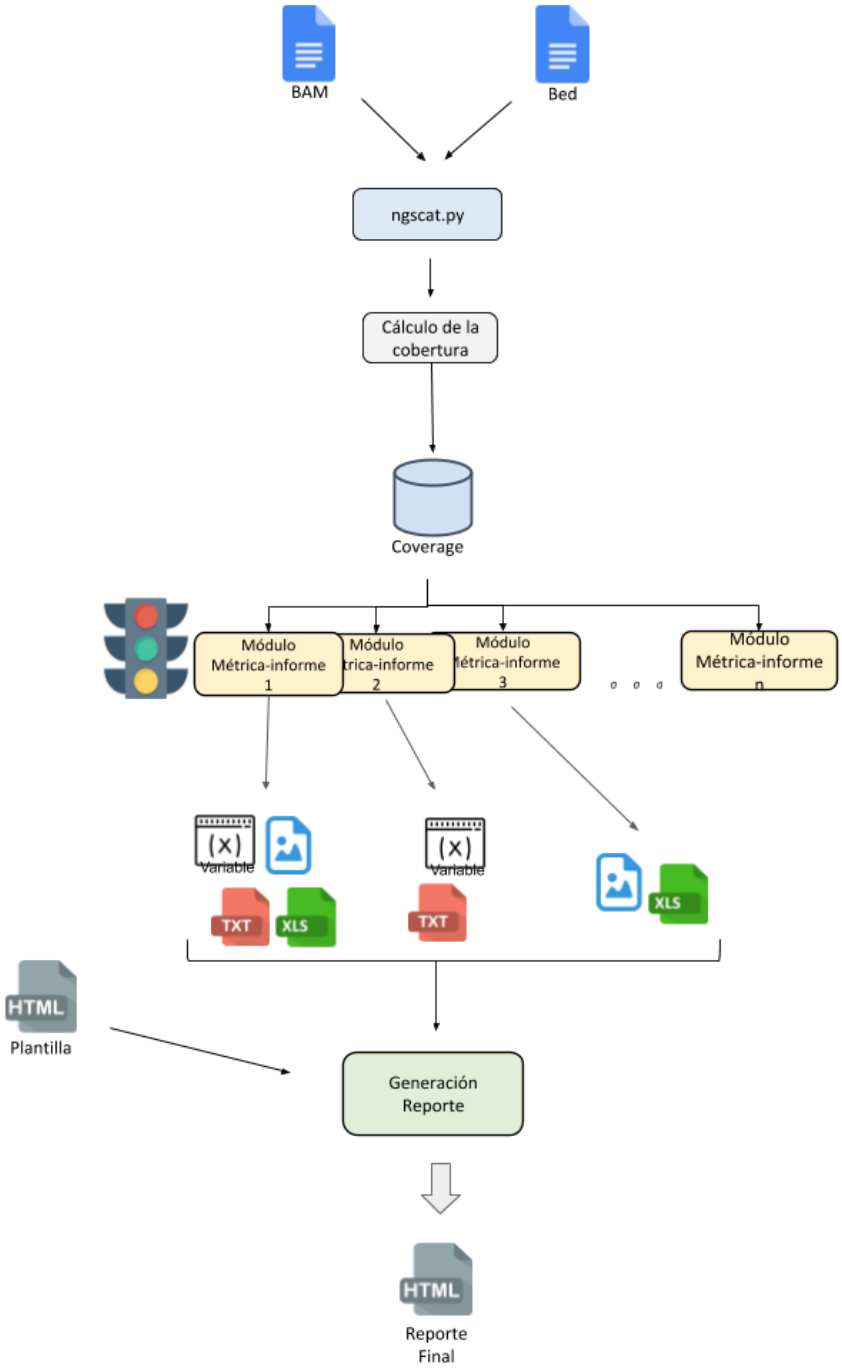

Figura 2.13: Flujo de trabajo de la herramienta *ngsCAT*. De entrada toma el archivo en formato BED y BAM, posteriormente son calculadas las profundidades/coberturas y son almacenados en un archivo en texto plano. A continuación se lanzan concurrentemente todas las métricas que leerán el archivo de profundidades y generarán los resultados. Cada módulo métrica-informe generará los resultados con diferentes formatos, como variables, imágenes, tablas o archivos de texto plano.

Tras la entrada del BAM y el BED se calculan las profundidades en las regiones de interés. En *ngsCAT* el proceso para calcular la profundidad en las diferentes regiones es *bam.mycoverage*. Este proceso escribe los datos de profundidad en un archivo de texto plano llamado *coverage* como podemos ver en la figura 2.13. Posteriormente cada uno de los procesos(métrica-informe) que dependen de las profundidades, leen del dispositivo de almacenamiento y almacenan campos de este archivo en memoria que cuando generan los resultados la liberan, así continuamente, leyendo de disco y almacenando en memoria para su posterior liberación. Como ya hemos comentado en la parte de flujo *ngsCAT*, para aprovechar todos los recursos de los procesadores con varios núcleos se ejecutan los diferentes módulos (métricas-informes) de una manera concurrente. En esta herramienta se recurre al uso de los semáforos que son una primitiva de sincronización y control de los procesos que ejecutan concurrentemente una sección de código. En este caso se lanzan todos los procesos y se bloquean utilizando semáforos y posteriormente el control de los semáforos va pasando por los diferentes módulos(métrica-informe). En *ngsCAT* el valor que toma el semáforo por defecto es 2 aunque es posible su cambio mediante el argumento *–nthreads*.

ngsCAT está diseñado como un conjunto de clases con sus respectivos métodos y una serie de funciones que calculan y generan los resultados de las métricas. Para ello, *ngsCAT* cuenta con cuatro clases localizadas en los scripts con su mismo nombre. Estas clases son:

- **bam_file()**: Es una clase heredada de la clase bam del paquete *pysam*, manteniendo por tanto los métodos de la clase padre y añadiendo métodos necesarios para el cálculo de métricas, generar gráficas o archivos en formato excel.

- **bed_file()**: La clase bed_file implementa métodos para la carga, comprobación del formato BED y la modificación de estos).

- **bedgraph_file()** la clase *BedGraph* contiene métodos para escribir y modificar archivos en formato *BedGraph*. El formato *BedGraph* es similar al BED a diferencia que en la primera linea tendremos una definición de la pista.(https://genome.ucsc.edu/goldenpath/help/bedgraph.html)

- **region()** permite manejar regiones cromosómicas

ngsCAT se dispone organizado en una carpeta con los archivos de código que componen el programa y otros directorios que contendrán la información necesaria para el correcto funcionamiento de la herramienta. En la carpeta raíz se encuentran todos los archivos de código, tanto el punto de entrada (archivo ngscat.py) como cada una de las métricas, que en este caso corresponde a cada proceso que lanza el programa concurrentemente. Por otro lado tenemos la carpeta *html* que contiene la plantilla que se usará para generar el informe final, y otras plantillas que serán utilizadas para integrar las métricas opcionales. Para generar el informe final también es necesaria la carpeta *img* que incluye los iconos e imágenes que serán empleados para generarlo (Figura 2.14).

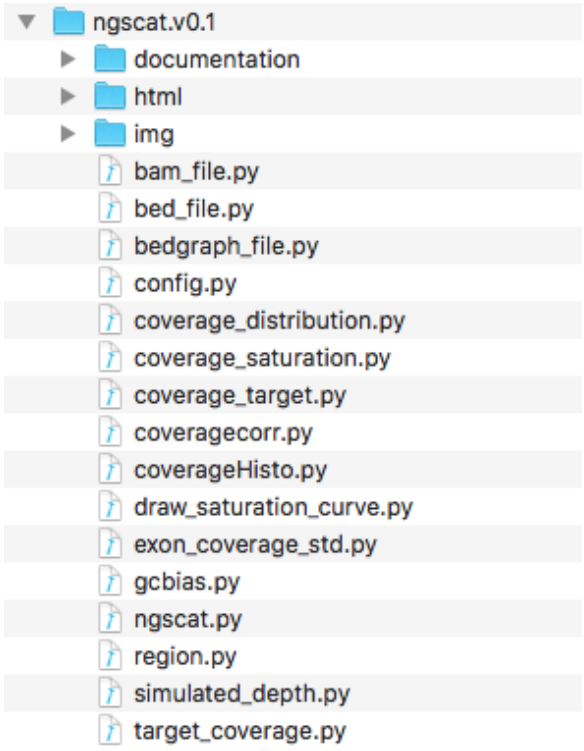

Figura 2.14: Organización de directorio *ngsCAT. En cada uno de los scripts se encuentra el código para el cálculo y la generación de los resultados de la métrica.*

Limitaciones de ngsCAT

El objetivo principal de este trabajo es mejorar la funcionalidad de la herramienta facilitar su uso y desarrollo futuro y mejorar la integración en otras herramientas de control de calidad. En *ngsCAT* existen ciertos aspectos que dificultan estas tareas. En esta sección veremos las principales características que limitan el fácil desarrollo de la herramienta así como factores limitantes en términos de uso.

- ***Mecanismo de lectura de datos***

  Como ya se ha descrito, la profundidad por base de cada una de las regiones de interés son almacenadas en un archivo de texto plano. Este archivo será leído y recorrido tantas veces como métricas que dependan de los datos de profundidad haya (Figura 2.13). Este mecanismo puede ser un factor limitante, ya que a medida que aumente el número de métricas aumentará el número de veces que se tienen que realizar operaciones de lectura del archivo donde se almacena la información de las profundidades en las regiones de interés. Además este mecanismo genera código redundante, ya que en cada módulo de métrica es necesario programar la lectura de este archivo.

- ***Control de concurrencia***

  La forma de control de concurrencia, donde todos los procesos son lanzados en conjunto y posteriormente bloqueados por semáforos cuyo control está distribuido en varios módulos (donde se calcula la métrica y se generan los resultados), dificulta la ampliación del código.

  Los semáforos son primitivas para la sincronización de bajo nivel que solucionan los problemas de compartición de recursos que presenta la concurrencia, sin embargo para su correcto funcionamiento es necesario que la gestión de este se distribuya por todo el código.

Dado el número de métricas que presenta la herramienta , cuando se utilizan los semáforos es complejo conocer cuantos procesos se ejecutan de una manera concurrente y cambiar el número de procesos que se ejecutan a la vez requiere modificar diferentes partes del código de todos los módulos, ya que como hemos comentado el control de estos semáforos está distribuido en cada una de las métricas. Además complica enormemente la adición de nuevas métricas, ya que el desarrollador tiene que realizar un seguimiento del control de los semáforos en los demás módulos.

- ***Estructura poco flexible***

  Como ya hemos visto, en la estructura actual los cálculos, la generación de las gráficas y la generación del informe HTML se realiza de una forma conjunta (monolítica) y no modularizada en profundidad. Esto limita la flexibilidad del programa, ya que obliga al desarrollador a identificar y modificar gran parte del código para, por ejemplo, generar un informe conjunto con los resultados de las métricas en formato JSON en vez de en HTML. Este problema lo podemos visualizar dando el ejemplo de la métrica de la cobertura por cromosoma. En este caso, una única función llamada *print_coverage* en el archivo *coverage_target.py* calcula los puntos a representar, genera las gráficas y además genera un archivo de texto plano en formato txt con las regiones de interés no cubiertas.

- ***Dependencias***

  Aunque *Python2.7* es una versión muy utilizada por la comunidad de desarrolladores, va a dejar de estar mantenida a partir de Enero del año 2020 https://www.python.org/dev/peps/pep-0373/ y no habrá una versión 2.8 https://www.python.org/dev/peps/pep-0404/. Por otro lado algunas dependencias no están actualizadas como por ejemplo *pysam* que utiliza la versión 0.7, cuyas clases y métodos no son totalmente compatibles con las versiones actuales (v0.15).

- ***Gráficas estáticas***

  Las gráficas generadas en el informe dan una idea global del resultado de cada métrica pero no proporcionan información detallada. Por ejemplo, no es posible conocer el valor exacto que se ha tomado para generar la gráfica, obligando al usuario a acceder a las tablas-resumen de cada métrica.

A pesar del correcto funcionamiento y gran versatilidad de la herramienta, el uso de tecnologías como *Python* 2, gráficas no interactivas, el complejo mecanismo de concurrencia y una estructura no modularizada limitan en cierto modo su uso y sobretodo su desarrollo futuro. Este trabajo fin de máster está enfocado no solo a mejorar la funcionalidad de *ngsCAT*, si no a permitir una mayor facilidad de uso y desarrollo. Para ello, se ha creado una nueva versión llamada *ngsCAT2*, cuyos detalles veremos en el siguiente capítulo. (Capítulo 3)

3

ngsCAT2: Sistema, diseño y desarrollo

3.1. Introducción

Considerando los objetivos de este trabajo, en concreto, la adición de la nueva funcionalidad, la mejora del apartado gráfico, la mejora en la instalación y la actualización de la tecnología, la base del código de la herramienta *ngsCAT2* se ha planteado como una evolución de la herramienta anterior. Tras el análisis de *ngsCAT* pudimos observar características limitantes como: la falta de modularidad, el diseño del mecanismo de concurrencia distribuido y el gran acoplamiento del código. Todas estas características hacen de *ngsCAT* una herramienta compleja de desarrollar lo cual dificulta la ejecución de los objetivos propuestos en este trabajo. Por ello, antes de la aplicación de los objetivos se han aplicado técnicas de refactorización que nos han permitido estructurar e independizar las distintas funciones facilitando la adición de nuevas métricas, la generación de salidas en otros formatos y la adición de gráficas interactivas.

A continuación vamos a realizar un recorrido sobre el nuevo diseño, implementación y la adición de nuevas funcionalidades en *ngsCAT2*.

3.2. Refactorización

Como se ha comentado en la introducción de este capítulo el proceso utilizado para solventar los problemas de diseño de *ngsCAT* es la refactorización. La refactorización (del inglés *refactoring*) es una técnica de la ingeniería de *software* que se puede definir como el proceso de cambiar un *software* de forma que no altera el comportamiento externo pero mejora la estructura interna. Esta mejora hace el *software* más fácil de comprender, facilita el control de errores (*bugs*) y permite el desarrollo de la aplicación utilizando menos tiempo y esfuerzo (33). El nuevo diseño de la herramienta se ha basado principalmente en los principios de diseño de separación de intereses (*concerns*), de responsabilidad única y en la búsqueda de una simplificación del mecanismo de concurrencia.

En el principio de responsabilidad única se establece que cada módulo o clase debe tener una única responsabilidad en la funcionalidad del software, y toda la responsabilidad debe estar encapsulada en la clase. El otro principio llamado principio de separación de intereses es un principio de diseño que establece la separación de un programa informático en secciones

distintas, en la cual cada sección desarrolla una función determinada e independientemente de las demás, ya que uno de los objetivos de este principio es la eliminación de interrelaciones entre secciones cuyo objetivo no es el mismo. Ambos principios han sido aplicados en el proceso de refactorización para reducir el acoplamiento (interdependencias) y mejorar la cohesión de los módulos del programa, permitiendo añadir nuevas métricas y nuevos formatos de salida de una manera más rápida y sencilla.

3.3. Diseño e implementación

Analizaremos en primer lugar de una forma general el diseño en el que se ha basado la nueva versión de la herramienta para posteriormente explicar con más detalle cada una de las partes donde se han realizado cambios, la justificación de ellos y algunos ejemplos de implementación.

Si comenzamos con el flujo normal de la herramienta, una de las limitaciones de *ngsCAT* (sección 2.3.2) es la falta de escalabilidad provocada por la lectura repetida del archivo que contienen las profundidades. Para solucionarlo, en *ngsCAT2*, se realizado un cambio en el diseño del flujo de los datos donde el archivo de las profundidades pasa a ser cargado en memoria.

Por otro lado, *ngsCAT* presentaba una falta de modularidad ya que cada uno de los módulos de programación calculaban la métrica y generaban directamente los resultados en diferentes formato. En *ngsCAT2* gracias a las aplicación de técnicas de refactorización se ha realizado una separación entre el cálculo de las métricas y la generación de los resultados. Esta separación de las métricas y la generación de los informes (resultados), como veremos más adelante, permite la reutilización del código que realiza los cálculos de las métricas para, por ejemplo, generar informes con diferente formato, como un nuevo informe con formato JSON. Aprovechando la separación del cálculo de las métricas y la generación de los resultados, se ha diseñado una estructura de generadores de informes, gracias a la cual, ha sido posible la adición de nuevos formatos de informe sin la necesidad de modificar diferentes partes del código.

Por otra parte, en *ngsCAT2* se ha introducido un nuevo diseño de paralelización o concurrencia de las métricas. Este nuevo diseño basado en *pooling* de procesos (podriamos denominarlo como reservorio de procesos), evita la utilización de sistemas de sincronización de bajo nivel como los semáforos para la gestión de los procesos, que como ya hemos comentado en la sección de limitaciones (sección 2.3.2) dificulta la adición de nuevas métricas.

Finalmente dada la separación entre métrica y generadores de informe, para la sincronización de ambos se ha optado por el uso de un principio de diseño llamado patrón observador, que permite ejecutar a los generadores de informes una vez que haya finalizado el cálculo de la métrica correspondiente.

Una vez comentadas las principales características del nuevo diseño vamos a describir en profundidad cada uno de los cambios.

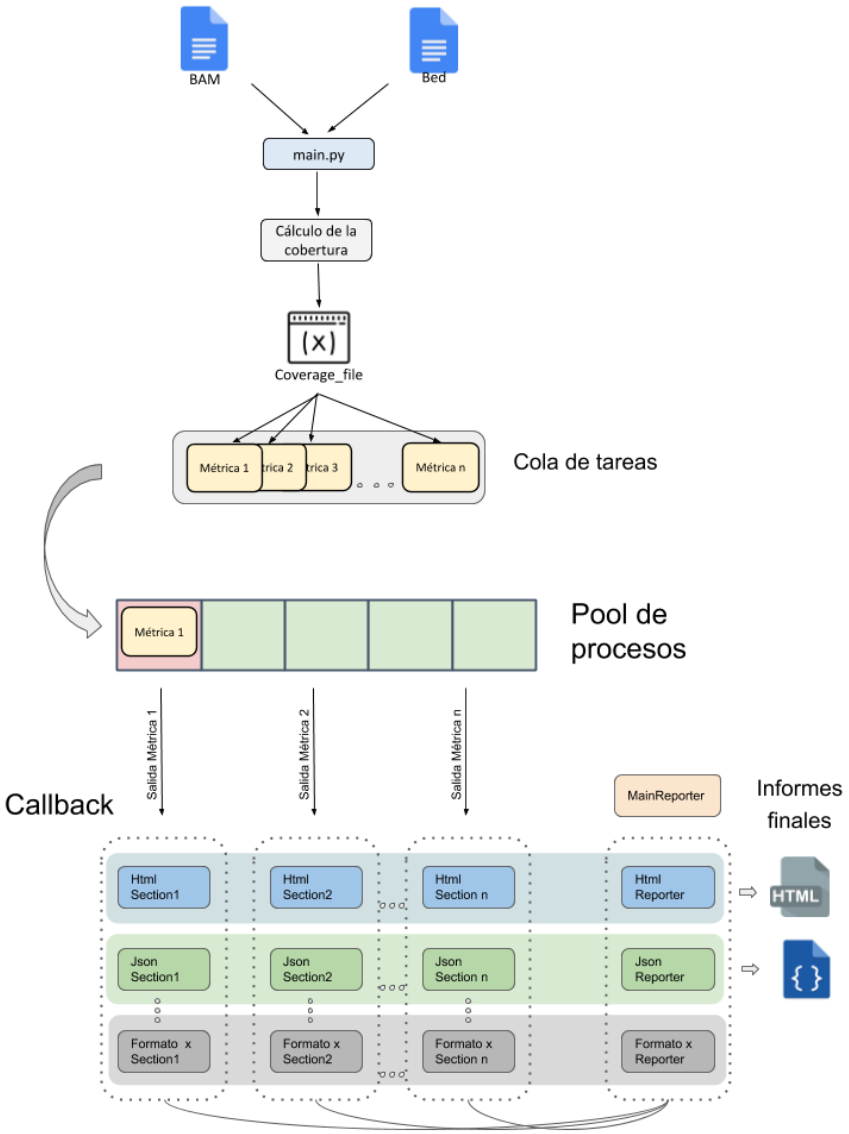

Figura 3.1: Diseño del flujo de la herramienta *ngsCAT2*. Para comenzar se toman los archivos de alineamientos y de regiones de interés (archivos BAM y BED respectivamente). Esta entrada se realiza a partir del *script* principal, llamado main.py. Posteriormente al igual que en *ngsCAT* se realiza el cálculo de los datos de profundidad. A diferencia de *ngsCAT*, estos datos son almacenados en la memoria RAM en la nueva versión de la herramienta. Posteriormente son añadidas a la cola de tareas del pool de procesos todas las métricas a calcular. Estas métricas serán sacadas de la cola y calculadas en los procesos disponibles del *pool* (fondo en color verde). Tras finalizar el cálculo de cada métrica son ejecutados los generadores de sección asociados a ellas. Como podemos observar cada color corresponde con un formato de informe: azul en formato HTML, verde en formato JSON, y gris formato x (cualquier formato) ya que para generar informes en otro formato se debe de seguir la misma estructura. Finalmente las secciones generadas de cada uno de los formatos son añadidos a los generadores de informe finales (Formato x Reporter) que las unen y conforman los informes finales)

3.3.1. Diseño de flujo de datos

Para solucionar la limitación de escalabilidad presente en *ngsCAT* que es producida por la lectura del archivo de profundidades *coverage* por cada métrica (sección 2.3.2) se han barajado dos posibles aproximaciones: La primera de ellas es la estrategia de procesamiento en flujo o *streaming*, en esta estrategia se toma el archivo (en este caso el archivo que contiene las profundidades) y se va leyendo en bloques. Los bloques de datos se van pasando a las diferentes métricas que con estos datos realizan los calculan y posteriormente los bloques se desechan. La principal ventaja de esta estrategia es el bajo requerimiento de memoria RAM, ya que el archivo *coverage* no es cargado completamente en esta. Sin embargo, complica notablemente la adición de nuevas métricas por parte de los desarrolladores, ya que es necesario coordinar todas las métricas para que vayan realizando los cálculos correspondientes a medida que se va recorriendo el archivo *coverage*. La otra aproximación es la lectura del archivo completo y la carga de éste en la memoria RAM. La principal ventaja de esta alternativa es que simplifica el acceso a los datos de profundidad por cada una de las métricas. Por otro lado la mayor desventaja a la que nos enfrentamos es un aumento del consumo de la memoria RAM.

Finalmente y aunque las dos opciones solventan el problema presente en *ngsCAT* ya que el archivo solo se lee una única vez, para el desarrollo de la herramienta se ha optado por la segunda opción, es decir, la carga completa en memoria RAM de los datos de profundidad. La base de esta decisión es que esta estrategia es más sencilla de implementar que facilitará el posterior desarrollo de la herramienta. Su principal desventaja que es el consumo de RAM no va a ser limitante puesto que el tamaño de los datos de profundidad de un exoma humano tiene un acotamiento aproximado de 8Gb. Esto es debido a que el número de bases de un exoma humano está determinado y que la forma de almacenar la información de profundidad en la memoria RAM donde unicamente se indica la la profundidad por cada base cubierta.

3.3.2. Diseño de la concurrencia

Al igual que en la herramienta anterior, en *ngsCAT2* se lanzan concurrentemente los módulos que calculan las métricas. El control de la concurrencia mediante semáforos de la herramienta *ngsCAT*, como hemos visto en la sección 2.3, dificultaba la adición de nuevas métricas ya que el control de los procesos se encontraba disperso entre los diferentes módulos. En *ngsCAT2* para simplificar el control de la concurrencia y concretamente para permitir un control más localizado y fácil de entender se ha optado por el uso de un *pool* de procesos.

El *pool* de procesos es un mecanismo de control de concurrencia de alto nivel donde se indica el número total de procesos o (procesos trabajadores) que se pueden ejecutar de una manera concurrente. A este *pool* se le indican las tareas que deseamos correr concurrentemente y se añaden a una cola de trabajos. En *ngsCAT2* las tareas se corresponden con cada una de las métricas (el cálculo de ellas). Tras esto, desde un punto de vista abstracto se puede decir que el mecanismo de *pool* gestiona automáticamente la eliminación de la métrica de la cola, la reserva del proceso necesario para su ejecución, el lanzamiento de la tarea (métrica) y la posterior liberación del proceso una vez la tarea haya finalizado.

Este *pool* de procesos está representado en la figura 3.1 por una caja con diferentes zonas, que serán los procesos trabajadores. Arriba del conjunto de procesos encontramos las métricas a ejecutar que se encuentran en cola, esperando a ser tomadas por alguno de los procesos libres. En el ejemplo, el *pool* de procesos cuenta con cinco procesos, es decir que nunca se van a realizar el cálculo de más de cinco métricas concurrentemente. El color rojo indica que ese proceso se encuentra ocupado por una de las métricas (en ejecución) y en verde los procesos que se encuentran libres.

Como acabamos de ver, la simplicidad que proporciona este mecanismo de alto nivel donde el control de todos los procesos se realiza en un único punto y cuya gestión se realiza automáticamente permite un fácil desarrollo de nuevas métricas ya que no es necesario la comprensión del complejo mecanismo de semáforos de la herramienta anterior, ni la modificación de diferentes módulos por parte del desarrollador. El desarrollador que desee correr una nueva métrica, únicamente se tiene que preocupar en indicarle al *pool* cuál es la función que debe de ejecutar.

Para la implementación de este mecanismo en *ngsCAT2* se ha hecho uso de la librería *multiprocessing*. En concreto el método utilizado para ello es *pool.apply_async*. Además con el objetivo de aprovechar al máximo la CPU el número de trabajadores por defecto se ha establecido como el número total de núcleos del procesador -1, lo cual deja un núcleo libre para el funcionamiento normal del sistema operativo.

3.3.3. Generadores de informes

Otro de los grandes cambios realizados en *ngsCAT2* es el nuevo diseño de los generadores de informes. Como vimos, en la sección de limitaciones (sección 2.3.2) en *ngsCAT* cada uno de los módulos de métrica generaba directamente las salidas en los diferentes formatos y la generación del informe HTML se realizaba de una manera conjunta. Esta estructura se salta el principio de diseño de separación de intereses y entorpece al desarrollador tareas como la adición de nuevas métricas o la generación de informes en otro formato. Por ello, en *ngsCAT2* aprovechando el nuevo diseño de informe dividido en secciones y a la separación entre métricas y generación de los resultados se ha creado una estructura jerarquizada de los generadores de informes.

La unidad mínima es el generador de informe de sección representado como *(Formato)Section* en la Figura 3.2, que no es más que una clase que toma los datos generados por una métrica determinada y los utiliza para generar la sección correspondiente dentro del informe final. Como la información generada por cada métrica se puede representar en diferentes formatos, cada métrica tendrá asociado un conjunto de generadores de informe de sección según el formato de las salidas que generen. Subiendo un paso más arriba en la jerarquía, encontramos a los generadores de informe de cada tipo de salida representados como *(Formato)Reporter*. Los generadores de informe *(Formato)Reporter* contendrán a todos los generadores de sección del mismo formato para, con ello, conformar el informe final. Finalmente subiendo un escalón en la jerarquía tenemos la clase que agrupa a todos los generadores de informes finales representado por *Reporter*. La clase *Reporter* se ha creado para lanzar cómodamente a todos los demás generadores de informes finales.

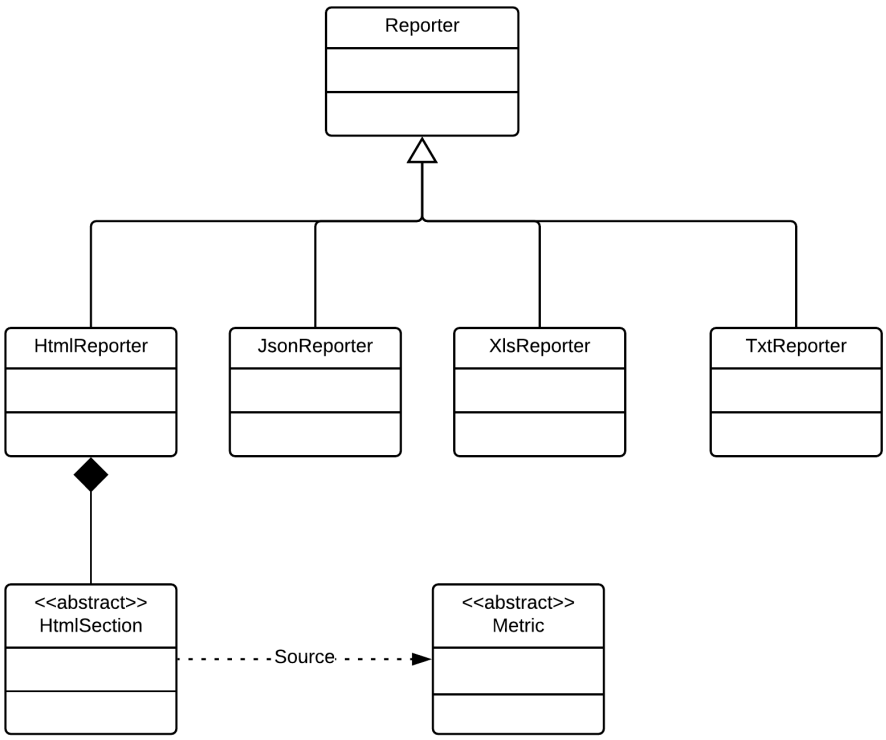

Figura 3.2: Gráfica del diseño de los generadores de informes. De arriba a abajo, tenemos una clase generadora global (*Reporter*) que agrupa a todos los generadores de informes de cada formato (*HtmlReporter,JsonReporter,XlsReporter...*). Donde cada uno de ellos está compuesto por generadores de sección del formato correspondiente. En este caso el generador de informe HTML está compuesto por todas las secciones de Html *HtmlSection* asociados a las métricas (*Metric*).

3.3.4. Patrón Observador para los informes

Como hemos visto, todos los generadores de sección asociados las métricas toman los datos que calcula la métrica, los procesan y lo añaden al generador de informe final correspondiente. Para realizar esta tarea es necesaria una sincronización entre la finalización del cálculo de las métricas y la generación de las secciones asociadas, ya que no se pueden generar las secciones sin los datos calculados por las métricas. Para solucionar esto se ha aplicado un patrón observador, cuya intención es definir las dependencias entre objetos para que, en el momento que un objeto cambie, el otro sea notificado.

Este patrón observador sigue el esquema presente en la figura 3.3 donde tenemos un sujeto que en nuestro caso son las diferentes métricas y posteriormente todos los observadores concretos que serán los generadores de sección asociados a la métrica. Como vemos la conexión entre observador concreto (generadores de sección) y sujeto no se realiza directamente. En medio de ellos existe la clase Observadora, que es la que se encarga de lanzar a todos los observadores concretos.

En *ngsCAT2* usando el mismo esquema, todos los generadores de sección (Observador-Concreto) asociados a una métrica son englobados en una clase llamada *CompoundReporter* (Observador). Para poder realizar esto, el conjunto de generadores de sección de cada métrica se han diseñado para que tomen la misma entrada (la salida de la métrica) y se ejecuten con el mismo método, para así poder ser llamados de una forma conjunta. Al diseño anterior, en términos técnicos, se le denomina interfaz. Esta idea también se encuentra representada en la

parte de abajo de la figura 3.1 por una linea de puntos que engloba a los generadores de sección asociados con la métrica 1.

En *ngsCAT2* se ha implementado este patrón haciendo uso de devolución de llamada o *callbacks* proporcionados por la librería *multiprocessing*. A estos *callbacks* se les pasa como argumento la función que deben de ejecutar tras la finalización del cálculo de cada una de las métricas, que en el caso de *ngsCAT2* es la generación del informe de sección determinado. Esto permite que se genere cada sección de los informes conforme finalice el cálculo de cada métrica sin más intervención, evitando así, agregar puntos de sincronización explícitos, que complicarían el flujo y posiblemente entorpecerían el paralelismo. Además permite el desacople entre métricas e informes, ya que el código de las métricas no tiene por qué estar relacionado directamente con los generadores de informes.

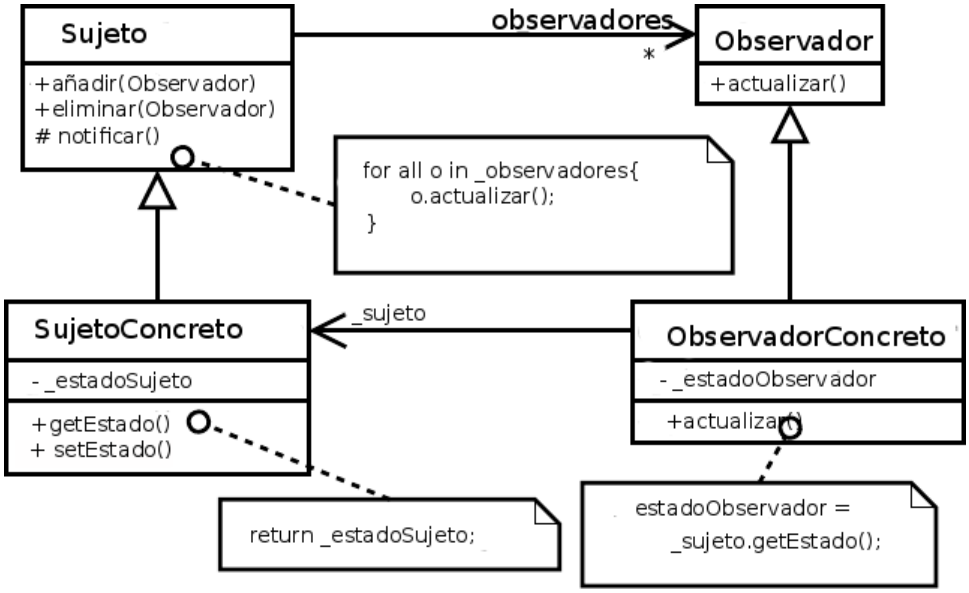

Figura 3.3: Estructura de patrón de diseño observador

En definitiva, en esta sección hemos explicado como, partiendo de la herramienta *ngsCAT*, en la cual habíamos detectado una alta interdependencia del código que dificultaba el desarrollo evolutivo, y aplicando técnicas orientadas a la refactorización destinadas a reducir el acoplamiento y a aumentar la cohesión, hemos obtenido una herramienta con la misma funcionalidad pero una mejor estructura. Esta mejora en la estructura facilita la mejora continua de la herramienta, como por ejemplo, la introducción de nuevas métricas sin la necesidad de modificar código de otras métricas o informes ni tampoco modificar la gestión de la concurrencia o añadir informes en otros formatos de salida sin la necesidad de modificar el código de cada una de las métricas.

A pesar de que han consumido la mayoría del tiempo de este trabajo y que no eran necesarias para el funcionamiento de la herramienta, todas estas mejoras de diseño se han considerado necesarias no solo por que han permitido la inclusión de las nuevas funcionalidades propuestas en este trabajo sin la necesidad de modificar el código original, si no porque se ha considerado que esta nueva estructura permitirá un desarrollo más rápido y simple de la herramienta por parte de desarrolladores externos, garantizando así la continuidad de esta.

3.3.5. Otras cuestiones de implementación

Con el objetivo de atraer al máximo número de desarrolladores y de prolongar la vida útil de la herramienta, para la implementación del diseño comentado se ha utilizado *Python* 3. *Python* 3 es la revisión más reciente de *Python* y la que continuará manteniendo el lenguaje ya que como se ha comentado anteriormente *Python* 2 va a dejar de estar mantenida a partir de 2020. Además se han actualizado las dependencias a las últimas versiones disponibles.

Además de las clases destinadas para el cálculo de métricas y las que conforman la estructura de generadores de informes, para el funcionamiento de *ngsCAT2* la herramienta cuenta con un conjunto de clases transversales que son usadas como base por el resto del código. Estas clases transversales son:

- **Bam_file:** Al igual que en ngsCAT, se trata de una clase heredada de la dependencia *pysam* que nos permitirá realizar cálculos tomando como entrada los ficheros de alineamientos (BAM) y de regiones de interés en formato BED
- **Bed_file:** Esta clase toma el archivo en formato BED y contiene métodos para su lectura y chequeo.
- **Coverage_file:** Clase implementada en esta nueva versión, con el objetivo de facilitar el acceso a los datos de profundidad. Además contiene métodos que permiten iterar y obtener datos contenidos en el objeto, permitiendo encapsular y jerarquizar todos los datos de profundidad que en *ngsCAT* se escribían en el archivo *coverage* para su uso. Agrupa a un conjunto de objetos de la siguiente clase:
  - **Chromosome:** Clase generada para almacenar todas las regiones que contiene un cromosoma. Habrá tantos objetos *Chromosome* como cromosomas en nuestro fichero de regiones de interés (formato BED). A su vez cada uno de los objetos de esta clase estará compuesto por un conjunto de objetos de tipo *Region*.
    - **Region:** Clase que tiene como atributos las especificaciones de la región. Estos atributos son el inicio y fin de la región en el cromosoma, la profundidad por base de la región, la media de estas y la desviación estándar. Habrá tantos objetos como regiones haya dentro del cromosoma.

A nivel de directorios *ngsCAT2* está organizado de una manera completamente diferente a la versión anterior. Partimos desde la carpeta raíz del programa (Figura 3.4), donde encontraremos un módulo llamado setup.py que contendrá el código necesario para la instalación del paquete. Además tenemos un conjunto de ficheros como MANIFEST.in que le indica a setup.py donde están los archivos necesarios, README.md contiene las instrucciones para instalar el paquete y un archivo de texto plano llamado *requirements.txt* donde tenemos un resumen de todas las dependencias que tiene el programa.

La carpeta ngscat2 contiene todo el código e información de la que se compone el programa. Dentro del directorio ngscat2 tenemos un par de ficheros y un conjunto de carpetas. Los ficheros son *main.py*, nuestro punto de entrada del programa y el que lanza todas las tareas y *config.json* en el cual están indicados argumentos para ajustar los umbrales que toman las métricas o modificar la representación de las salidas (figura 3.4).

Por otro lado los módulos o directorios *html* e *img* contienen las plantillas en formato HTML usadas para generar el informe final y las imágenes necesarias para la generación de este respectivamente.

En el módulo (directorio) *misc* tenemos un conjunto de *scripts* que contienen las clases transversales (Bam_file, Bed_file, Coverage_file, Chromosome, Region) las cuales se han indicado anteriormente.

Posteriormente cada uno de las métricas y generadores de informes se encuentran en el directorio *modules*. Dentro de este encontramos el directorio *metric* que contendrá el código que calcula las diferentes métricas y la carpeta *report* que contiene a todos los generadores de informes.

La carpeta *metric* está compuesta de otros directorios o módulos con el nombre de la métrica correspondiente. Dentro de cada uno de estos directorios se encuentra un *script* llamado metric.py que contiene la clase con los métodos necesarios para calcular la métrica correspondiente como podemos ver en la zona central de la figura 3.4.

Por otro lado en la carpeta *report* como vemos en la figura 3.4 tenemos ordenados por tipo de formato los diferentes generadores de informes. Dentro de cada formato de informe tenemos las carpetas con el nombre de las métricas asociadas, que a su vez contendrán el *script* report.py que contiene el generador de sección de informe de la métrica correspondiente (indicado en el nombre del directorio).

Asi pues cada métrica tendrá una carpeta dentro del directorio *metrics* y un *script* llamado metric.py donde estará el código de ella, por ejemplo la métrica de cálculo de la distribución (depth_distribution). Posteriormente en la carpeta *report*, si se ha determinado que los resultados de la métrica se generen en formato HTML, dentro de la carpeta *html* habrá un carpeta con el nombre de la métrica. Y en ella tendremos un *script* llamado report.py que contiene el código necesario para generar la sección del informe de la métrica de distribuciones en el formato indicado. Si los datos calculados son representados con otro formato, encontraremos dentro de las carpetas de formato una carpeta con el nombre de la misma métrica que contendrá el código para generar la sección en otro tipo de formato.

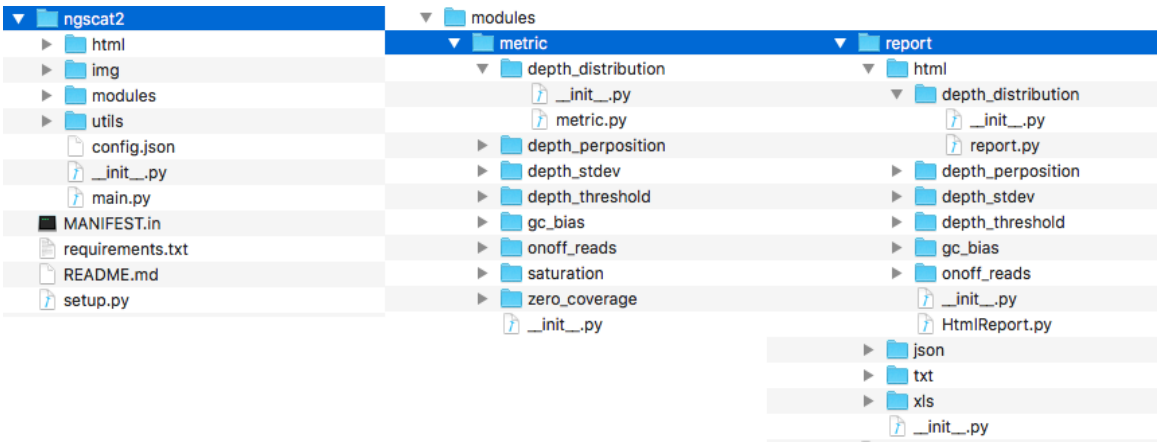

Figura 3.4: Esquema de organización de los directorios de *ngsCAT2*. A la izquierda vemos la carpeta raíz de la herramienta, en el centro vemos los módulos dentro de *modules metrics*, cada una carpeta se corresponde con un módulo métrica. A la derecha vemos el contenido de la carpeta *report* que contiene, organizado por tipo, a todos los módulos que generan las diferentes secciones.

3.4. Nueva funcionalidad, mejoras adicionales de la herramienta

En esta sección, veremos las nuevas funcionalidades añadidas, los cambios realizados en las métricas ya existentes y otras características de esta versión.

3.4.1. Anotación de regiones

En esta nueva versión se ha añadido una nueva funcionalidad que nos permite anotar todas las regiones presentes en los archivos .bed de salida, ahorrando al usuario de la herramienta la anotación de forma manual de cada uno de ellos. Esta función nos permite conocer, por ejemplo, qué genes se encuentran en nuestras regiones de interés sin cobertura e incluso en qué zonas de los respectivos genes se localizan (figura 3.5). Además de la anotación de las regiones sin cobertura también son anotadas las regiones de interés cubiertas, en ese caso, en el archivo en formato BED de salida está representado la región cromosómica, la profundidad y a qué gen y zona pertenecen. Estas zonas pueden ser regiones exónicas, intrónicas o regiones UTR 5/3' como podemos ver en la figura 3.6. Para la anotación de las regiones es necesario indicarle al programa un archivo en formato BED con regiones cromosómicas previamente anotadas, este archivo tiene que tener un formato determinado (Figura 3.7). A parte del exoma humano anotado, *ngsCAT2* permite el uso de archivos BED de anotaciones de otros organismos o ensamblados. Finalmente, la métrica de anotación se trata de una métrica opcional y para el correcto funcionamiento de ella es necesario tener instalada la dependencia *BEDOPS* (34).

```
chr4  145788  145789  NM_001289930_intron_0_0_chr4_125180_f;NR_110528_intron_0_0_chr4_124556_f;NR_110529_intron_0_0_chr4_124556_f
chr4  148781  148981  NM_001289930_intron_0_0_chr4_125180_f;NR_110528_intron_0_0_chr4_124556_f;NR_110529_intron_0_0_chr4_124556_f
chr4  150029  150229  NM_001289930_intron_0_0_chr4_125180_f;NR_110528_intron_0_0_chr4_124556_f;NR_110529_intron_0_0_chr4_124556_f
chr4  154645  154649  NM_001289930_intron_0_0_chr4_125180_f;NR_110528_intron_0_0_chr4_124556_f;NR_110529_intron_0_0_chr4_124556_f
chr4  159119  159124  NR_110528_intron_0_0_chr4_124556_f
chr4  159472  159473  NR_110528_intron_0_0_chr4_124556_f
chr4  166506  166707  NR_110528_intron_0_0_chr4_124556_f
chr4  183127  183327  NR_110528_intron_0_0_chr4_124556_f
chr4  184147  184358  NR_110528_intron_0_0_chr4_124556_f
chr4  206468  206669  NR_027481_intron_0_0_chr4_206571_f;NR_027481_utr5_0_0_chr4_206389_f
chr4  212403  212604  NR_027481_intron_0_0_chr4_206571_f
chr4  213030  213230  NR_027481_intron_0_0_chr4_206571_f
chr4  213606  213807  NR_027481_intron_0_0_chr4_206571_f
chr4  247853  247854  NR_027481_utr5_1_0_chr4_247349_f
chr4  248692  248694  NR_027481_utr5_1_0_chr4_247349_f
chr4  264523  264764  NM_001137608_utr3_0_0_chr4_264464_r
chr4  289174  289374  NM_001137608_cds_1_0_chr4_289227_r;NM_001137608_intron_0_0_chr4_266420_r;NM_001137608_intron_1_0_chr4_289323_r
chr4  289781  289981  NM_001137608_cds_2_0_chr4_289818_r;NM_001137608_intron_1_0_chr4_289323_r
```

Figura 3.5: Ejemplo salida de regiones sin cobertura anotadas (NoCoverage.txt)

```
chr15 49135804  49135806 7  NM_203349_intron_2_0_chr15_49135786_r
chr15 49135806  49135807 6  NM_203349_intron_2_0_chr15_49135786_r
chr15 49135807  49135809 5  NM_203349_intron_2_0_chr15_49135786_r
chr15 49135809  49135811 4  NM_203349_intron_2_0_chr15_49135786_r
chr15 49135811  49135814 3  NM_203349_intron_2_0_chr15_49135786_r
chr15 49135814  49135815 2  NM_203349_intron_2_0_chr15_49135786_r
chr15 49170353  49170356 1  NM_014335_utr5_0_0_chr15_49170290_f;NM_203349_intron_7_0_chr15_49164340_r
chr15 49170356  49170357 3  NM_014335_utr5_0_0_chr15_49170290_f;NM_203349_intron_7_0_chr15_49164340_r
chr15 49170357  49170358 5  NM_014335_utr5_0_0_chr15_49170290_f;NM_203349_intron_7_0_chr15_49164340_r
chr15 49170358  49170361 9  NM_014335_utr5_0_0_chr15_49170290_f;NM_203349_intron_7_0_chr15_49164340_r
chr15 49170361  49170362 10 NM_014335_utr5_0_0_chr15_49170290_f;NM_203349_intron_7_0_chr15_49164340_r
```

Figura 3.6: Ejemplo de salida en formato BED con regiones cubiertas anotadas

```
chr1   11873  12227  NR_046018_utr5_0_0_chr1_11874_f  0  +
chr1   12227  12612  NR_046018_intron_0_0_chr1_12228_f  0  +
chr1   12612  12721  NR_046018_utr5_1_0_chr1_12613_f  0  +
chr1   12721  13220  NR_046018_intron_1_0_chr1_12722_f  0  +
chr1   13220  14409  NR_046018_utr5_2_0_chr1_13221_f  0  +
chr1   14361  14829  NR_024540_utr3_0_0_chr1_14362_r  0  -
chr1   14829  14969  NR_024540_intron_0_0_chr1_14830_r  0  -
chr1   14969  15038  NR_024540_utr3_1_0_chr1_14970_r  0  -
chr1   15038  15795  NR_024540_intron_1_0_chr1_15039_r  0  -
chr1   15795  15947  NR_024540_utr3_2_0_chr1_15796_r  0  -
```

Figura 3.7: Formato archivo entrada para la anotación de las regiones.

3.4.2. Figuras generadas por ngsCAT2

En *ngsCAT* una de las limitaciones en términos de usabilidad era la falta de información que proporcionaban las gráficas generadas en el informe (véase sección 2.3.2). En *ngsCAT2* para solucionar la limitación de las gráficas estáticas, se ha recurrido al uso de la librería *Plotly*.

Plotly (12) es una librería cuya función es la creación de gráficos interactivos. La librería está disponible en *Python* pero también en diversidad de lenguajes como *Javascript*, *MATLAB*, *Perl* y *R*. Esta diversidad nos ayuda a atraer a desarrolladores, en concreto a aquellos que se dedican al análisis de datos. Pero la principal ventaja de la adición de las gráficas interactivas es que permite al usuario una visualización de los valores exactos resultados de cada métrica. Además se puede mostrar información adicional como podemos ver en ejemplo de la figura 3.8. En ella podemos ver el *tooltip* o desplegable que aparece si posicionas el cursor encima de uno de los puntos, además se puede visualizar información sobre las regiones que componen ese punto como: su localización genómica de inicio y de fin, la media de la profundidad de las bases que componen a esa región y su desviación estándar. En la figura 3.9 podemos ver las gráficas generadas por *ngsCAT2* frente a *ngsCAT* del mismo experimento de secuenciación.

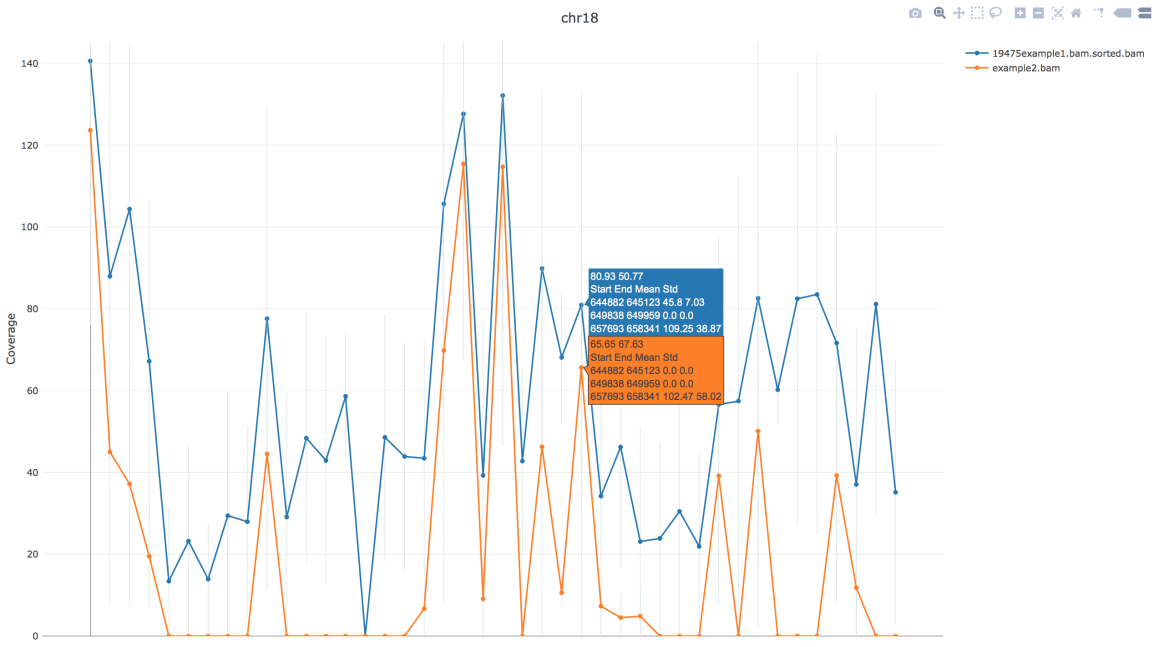

Figura 3.8: Ejemplo de gráfico interactivo proporcionado por *ngsCAT2*. En concreto una de las gráficas de la métrica de uniformidad de profundidad por cromosoma. Además se han introducido dos archivos de alineamiento (BAM). La interactividad permite ver la información de cada punto y además comparar el resultado de ambos experimentos

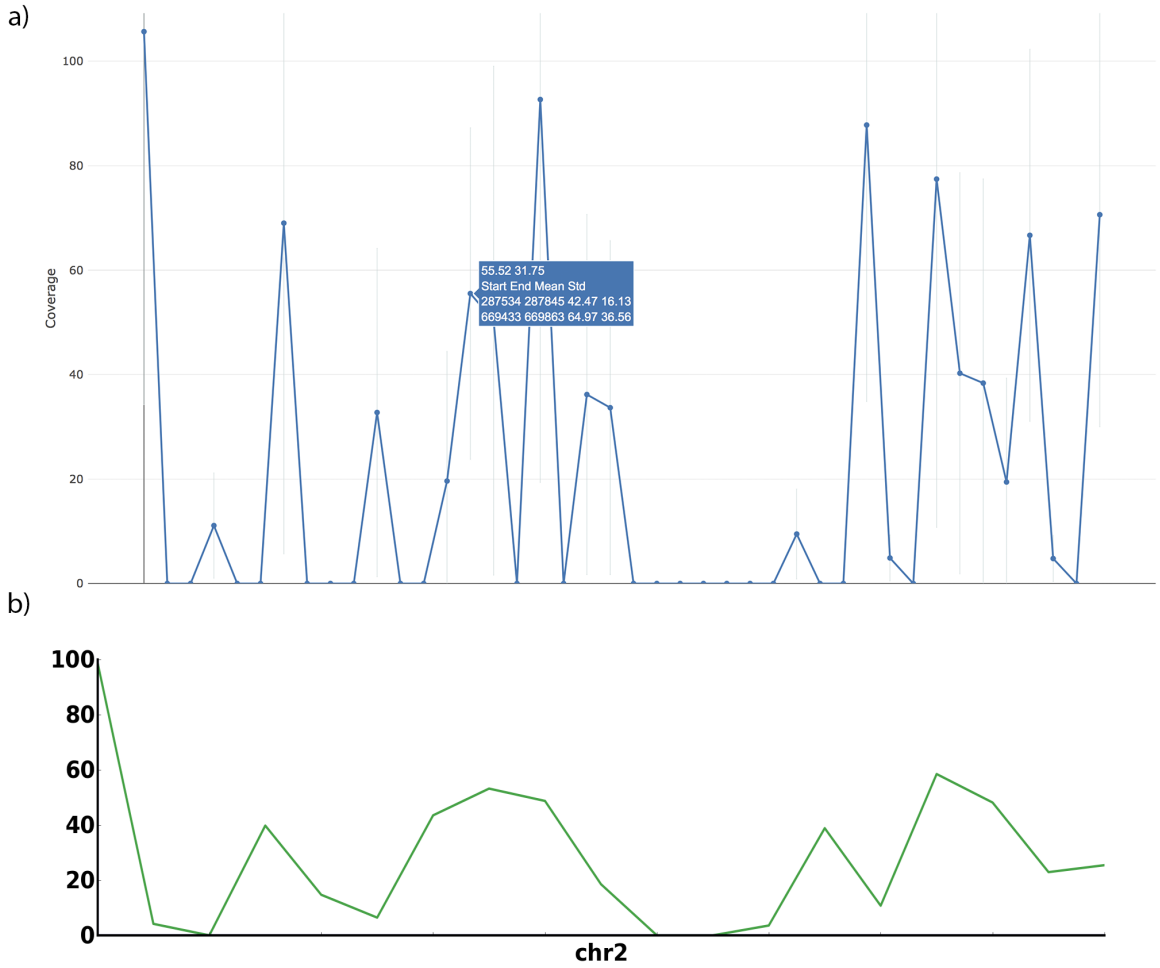

Figura 3.9: Comparación gráficas de profundidad por posición. a) Gráfica de profundidades del cromosoma 2 generada por *ngsCAT2*. b) Gráfica de profundidades del cromosoma 2 generada por *ngsCAT*, donde podemos observar las regiones contenidas en el punto representado y sus respectivas medias y desviaciones estándar.

3.4.3. Nuevos archivos generados por la herramienta

Por otro lado, aprovechando la nueva estructura de generadores de informes, se han añadido nuevos archivos de salida en formato JSON (JavaScript Object Notation, notación de objeto de javascript) https://www.json.org que es un formato de texto ligero diseñado para el intercambio de datos. Además es totalmente independiente del lenguaje, aunque usa convenciones que son familiares para los programadores. Estas propiedades hacen al JSON un formato ideal para el intercambio de datos y su utilización en esta herramienta permitirá a los desarrolladores el uso de los resultados de ella para su inclusión en otro tipo de herramientas.

En *ngsCAT2* se genera un archivo JSON por cada métrica que tenga como salida una tabla. Estos archivos JSON contienen todos los datos generados por la métrica, la información de los archivos de entrada, los parámetros utilizados y si los resultados se encuentran dentro de los umbrales definidos por el usuario (*"status"*). Un ejemplo de esta salida podemos verla en la figura 3.10, en concreto se trata de la métrica calculada por reads_on_target.

```
{
  "results": [
    {
      "bamfilename": "/home/agarcia/exampledata/example2.bam",
      "enrichment": 5069.30533989366,
      "onread": 159900,
      "percontotal": 67.31639547856105,
      "totalread": 237535,
      "onperchr": {
        "chr2": 2613,
        "chr3": 1627,
        "chr4": 11531,
        "chr5": 10105,
        "chr6": 2338,
        "chr7": 6732,
        "chr8": 1956,
        "chr9": 4592,
        "chr10": 5160,
        "chr11": 20660,
        "chr12": 6839,
        "chr13": 3320,
        "chr14": 5582,
        "chr15": 6387,
        "chr16": 24565,
        "chr17": 7022,
        "chr18": 3428,
        "chr19": 19112,
        "chr20": 7028,
        "chr21": 1272,
        "chr22": 4215,
        "chrX": 1905,
        "chrY": 1911
      },
      "totalperchr": {
        "chr2": 5125,
        "chr3": 1686,
        "chr4": 15826,
        "chr5": 11403,
        "chr6": 3526,
        "chr7": 8921,
```


Figura 3.10: En esta salida están representados los resultados generados por la métrica, como por ejemplo: el enriquecimiento obtenido "*enrichment*", el número de lecturas que cubren las regiones de interés "*onread*", también contiene información a nivel de cromosomas como el número de lecturas que cubren las regiones de interés 'onperchr'...

Una de las utilidades de estos JSON puede ser la integración de los resultados en *software* que englobe resultados de herramientas bioinformáticas. Uno de ellos es *MultiQC* (35) que es una herramienta que a partir de los resultados de diversas herramientas bioinformáticas de análisis de calidad (*FASTQC*, *fastp*, *picard tools*) permite generar un solo informe conjunto de control de calidad.

3.4.4. Control de versiones, GitHub

En el desarrollo de este proyecto se ha utilizado un sistema de control de versiones distribuido (DCVS) que permite a los desarrolladores trabajar de una manera distribuida y eficiente en el mismo proyecto de desarrollo de software. De todos los DCVS disponibles se ha optado por *Git* que es el sistema de control de versiones moderno más usado en el mundo. Para que el proyecto sea visible a la comunidad de desarrolladores se ha utilizado el servicio facilitado por la plataforma GitHub, que cuenta con más de 31 millones de usuarios y aproximadamente 100 millones de repositorios `https://github.com/about`. Gracias a esta plataforma cual-

quier usuario o desarrollador puede descargarse la herramienta *ngsCAT2* completa para su uso. Además esto permite un desarrollo colaborativo, donde los usuarios pueden clonar la herramienta y hacer sus propios desarrollos en diferentes ramas con la posibilidad de luego incorporarlos al proyecto. El repositorio creado en GitHub se puede acceder mediante el siguiente enlace https://github.com/babelomics/ngscat2.

3.4.5. Creación de paquete Python

En todo tipo de Software el proceso de instalación es un paso importante que afectará a la cantidad de usuarios de la herramienta. Por ello y con el objetivo de aumentar el uso por parte de usuarios no expertos se ha generado un paquete de *Python*, instalable mediante el sistema de gestión de paquetes *pip*. El paquete de *Python* contiene toda la información y permite la instalación automática de las dependencias de *Python* necesarias para el correcto funcionamiento de la herramienta. Además permite a los desarrolladores importar esta herramienta como módulo para su inclusión en nuevos proyectos. véase Manual de utilización A.

3.4.6. Otras consideraciones

A continuación describimos otras consideraciones importantes en la implementación de *ngsCAT2*.

Cambio Formato Archivo de Configuración

En *ngsCAT* los argumentos para la personalización del análisis se encuentran en el archivo *congig.py*, en cambio en *ngsCAT2*, pensando en su implementación en otras herramientas este archivo de configuración ha pasado a ser un archivo en formato JSON llamado *config.json*.

Corrección de errores o bugs

Gracias a el profundo proceso de refactorización de la herramienta *ngsCAT* se ha detectado un error en una métrica perteneciente a la sección de especificidad. En concreto éste se ha detectado en la métrica que aporta el número de lecturas duplicadas tanto dentro como fuera de las regiones de interés (*duplicates on/off target*). El error provocaba que no se sumaran las lecturas que aparecen mayores o iguales al último valor definido por el usuario, y ha sido arreglado en esta versión.

Modificación métricas

Posteriormente en la sección de uniformidad se ha añadido un nuevo argumento que permite modificar el numero de cajas o *bins* para representar el histograma. El argumento es *distributionbins* modificable en el archivo de configuración *config.json*.

Uno de los cambios más significativos, aprovechando la refactorización y la adición de gráficas interactivas, ha sido realizado en la métrica de uniformidad que mide la profundidad por posición (*coverage per position*). En esta métrica, se toma la lista concatenada de la profundidad por base en las regiones de interés y se realiza un agrupamiento de estos para poder representarlos y dar una idea de la cobertura a lo largo del cromosoma. En la herramienta anterior se establecía automáticamente el número de bases que conformaban cada uno de los puntos representados dividiendo el número de bases de todas las ROIs. Esto daba lugar a diferencias en la resolución entre cromosomas, ya que los cromosomas donde se localizaban menos regiones de interés presentaban una menor resolución (menor número de puntos). Además, en estos puntos podían estar representados fragmentos de una o varias regiones de interés.

Por ello, en esta versión se ha decidido calcular el número de puntos representados según el número de bases de las regiones de interés en cada cromosoma y no del total de todas las

bases de las ROIs. Con ello mejoramos la resolución, ya que en cada cromosoma tendremos el mismo número de puntos representados. Además, cada uno de los puntos representados están conformados por regiones completas no siendo posible que tome fragmentos de regiones de interés. Finalmente, en esta versión es posible el control del número de puntos representados por cromosoma, mediante la variable *npointsperchrom* presente en el archivo de configuración *config.json*.

4

Caso de Estudio

4.1. Introducción

En este capitulo se exponen los resultados de una ejecución completa de *ngsCAT2* con el objetivo de visualizar los cambios realizados en ella y para demostrar su correcto funcionamiento. Por otro lado se ha realizado una prueba de rendimiento comparativa en la cual se ha cuantificado tiempo de ejecución y consumo de memoria RAM comparándolo con la primera versión de la herramienta.

La muestra utilizada será la de un exoma completo, en este caso se ha descargado de la fase 3 del proyecto de los 1000 genomas, en concreto el archivo de alineamientos en formato BAM de la muestra HG00097 que tiene un tamaño de 5.8 Gigabytes, 57,270,738 lecturas *reads* y una longitud de las lecturas de 100 bases, descargado desde `http://ngscat.clinbioinfosspa.es/_media/ngscat/download/hg00097.bam`. El archivo de regiones de interés usado es el generado por el mismo proyecto, donde están indicadas todas las regiones codificantes, en concreto, se tratan de todas las regiones exónicas. La suma de las regiones cromosómicas de interés es de 46,492,725 de bases, este archivo en formato BED puede ser descargado de `ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/reference/exome_pull_down_targets/20130108.exome.targets.bed`.

4.2. Caso de estudio

Tras la instalación de la herramienta (ver Anexo Manual de uso A) se ha procedido a analizar el caso de estudio de exoma completo. Desde la terminal se ha utilizado el siguiente comando (cada de código 4.1) donde le indicamos la ruta absoluta del archivo de alineamientos en formato BAM, que ya se encuentra indexado, la ruta absoluta de nuestro fichero con las regiones de interés y, en este caso, le vamos a indicar un genoma de referencia (para poder generar la gráfica del sesgo en el contenido GC). Además para correr la nueva funcionalidad, se ha indicado un archivo en formato BED que proporciona la anotación del genoma humano (ensamblado 37) obtenida de UCSC (University California Santa Cruz).

```
ngscat2 --bam /home/agarcia/exomadata/hg00097.bam
--bed /home/agarcia/exomadata/exome.targets.g1k.bed
--reference /home/agarcia/exomadata/hg19.wochr.fa
--annotation /home/agarcia/exomadata/refgene_hg19.bed
--out /home/agarcia/pruebaexomalast2
```

Caja de código 4.1: Ejecución de ngsCAT2

Una vez ejecutado el programa en el directorio de salida indicado se genera la carpeta con los resultados (figura 4.1). Dentro de ella encontramos el informe final html con el nombre **CaptureQC.hmtl** y la capeta *data*, además la carpeta *img* y un archivo de estilos usados en la generación del informe. Dentro de la carpeta *data* encontramos todos los archivos generados por la herramienta, en ella se encuentran los HTML de cada una de las gráficas interactivas, las tablas XLS que nos proporcionan datos del resultado de las métricas, los archivos JSON que contienen la información presente en las tablas e información importante de las métricas. Además encontramos por cromosoma el archivo en formato BED que contiene la información de las profundidades dentro de las regiones de interés. El nombre de estos archivos es (nombredelamuestra.número_de_cromosoma.bed). Por otro lado, como se comentará en la siguiente sección se generan por cada cromosoma los archivos con la información de las regiones con alta profundidad pero fuera de las regiones de interés, el nombre de estos archivos es (nombredelamuestra.número_de_cromosoma.off.bed). El archivo *NoCoverage.txt* contiene las regiones de las regiones de interés que no están cubiertas por ninguna lectura. Además, como es el caso, gracias a la nueva funcionalidad se han anotado todos los archivos en formato BED generados, por ello antes de la extensión *.bed* se ha añadido la extensión *.annotated*.

El archivo que contiene los resultados de la ejecución de *ngsCAT2* con este ejemplo pueden descargarse de https://mega.nz/#!eywTGCBL!6HTDl7J9eLY4VhX2aFfe57Cp9mvwLw4JTcDcHr-zu1A

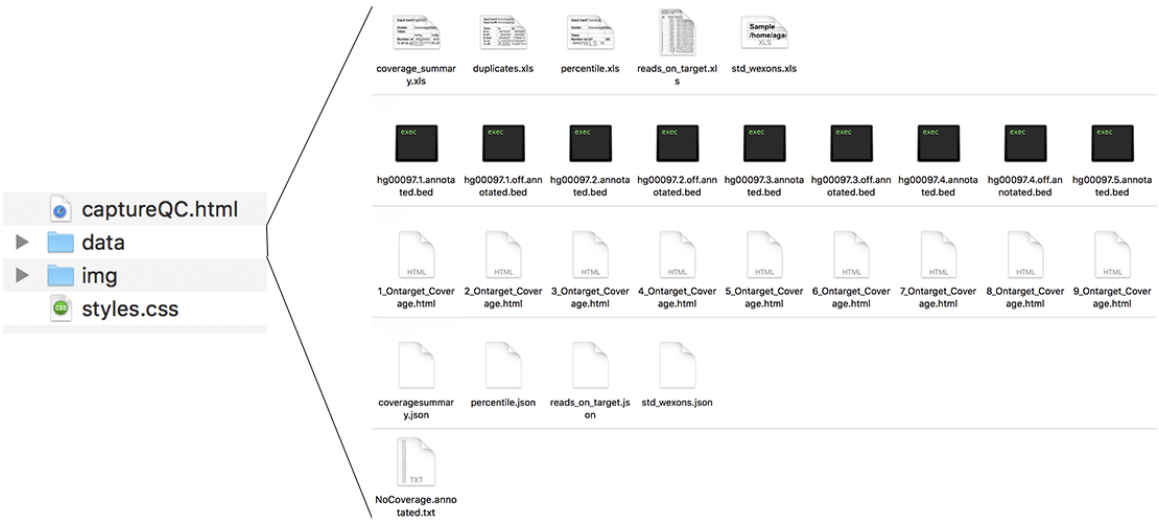

Figura 4.1: Esquema directorio de las salidas generadas por ngsCAT2. A la izquierda el contenido de la carpeta de resultados, a la derecha el contenido de la carpeta data

A continuación vamos a ver los resultados del análisis en términos de sensibilidad, especificidad y uniformidad, enfatizando las novedades aportadas por *ngsCAT2*.

4.2.1. Sensibilidad

Para comenzar vamos a ver cuáles son los resultados en término de sensibilidad. A modo de recordatorio, ya que lo vimos en la sección 2.3.1, *ngsCAT2* proporciona una métrica que permite saber cómo de bien están cubiertas las regiones de interés. Esta métrica es llamada porcentaje de bases cubiertas a diferentes umbrales (*Percentage of target bases covered at diferent coverage thresholds*). En un experimento de secuenciación dirigida común lo normal es que más de un 90 % de las bases de las ROIs se encuentren cubiertas por al menos una lectura. Como podemos ver en la figura 4.2, gracias a la adición de gráficas interactivas en la nueva versión de la herramienta podemos obtener el valor exacto representado en la misma gráfica. Entonces, si situamos el cursor en la barra de ≥ 1 vemos que un 97,27 % de las bases que conforman las regiones de interés están cubiertas por una o más lecturas además no hay una bajada considerable incluso a 30x de profundidad, lo cuál nos indica que en términos de sensibilidad el proceso de enriquecimiento ha sido exitoso. En *ngsCAT2* al igual que en *ngsCAT* se le puede indicar mediante el argumento *–coveragethrs* la lista de los umbrales a representar. El umbral en el cual salta un aviso se puede modificar mediante el argumento *"warnbasescovered"* del archivo *config.json* que por defecto se sitúa en 90.

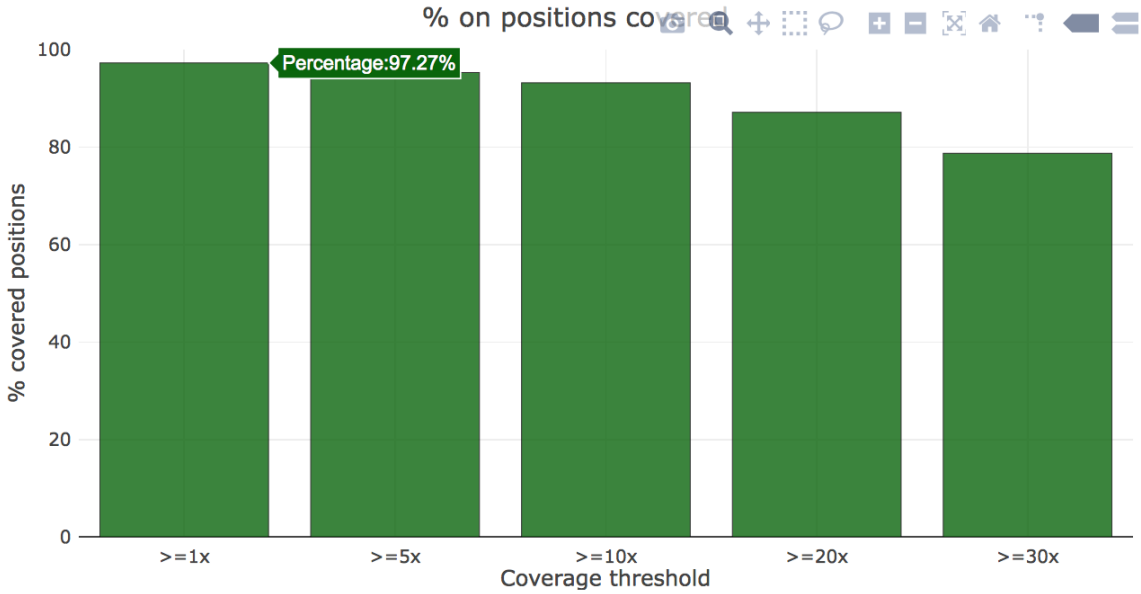

Figura 4.2: Porcentaje de bases cubiertas a diferentes umbrales.

4.2.2. Especificidad

En términos de especificidad *ngsCAT2* proporciona información que nos permite ver cuanto esfuerzo de secuenciación hemos utilizado en capturar zonas fuera de las regiones de interés. Esta sección se divide en dos partes. En la primera llamada *number of reads on target* o número de lecturas en la región de interés se generan varios archivos. Comencemos por la figura 4.3, en ella podemos ver el porcentaje de lecturas que cubren regiones de interés frente al total de lecturas que alinean en el cromosoma. A parte de la gráfica *reads on target*, se genera una tabla en formato XLS de nombre *reads_on_target.xls* que complementa los datos de que proporciona la gráfica interactiva.

Un resultado deseable sería que más de un 80 % de las lecturas cubran las regiones de interés lo cuál nos indicaría que el proceso de captura ha funcionado correctamente, y que no se ha malgastado mucho esfuerzo en la secuenciación de zonas que no nos interesan. En este caso la

media del porcentaje de lecturas en regiones de interés es menor, en el *summary* del informe podemos ver que es de un 76.2 % lo que nos indica que ha habido lecturas que han alineado fuera de nuestras regiones de interés.

En esta métrica podemos configurar el umbral mínimo del porcentaje de lecturas que cubren las regiones de interés mediante la variable, presente en el archivo de configuración, *"warnon-target"* que por defecto es 80.

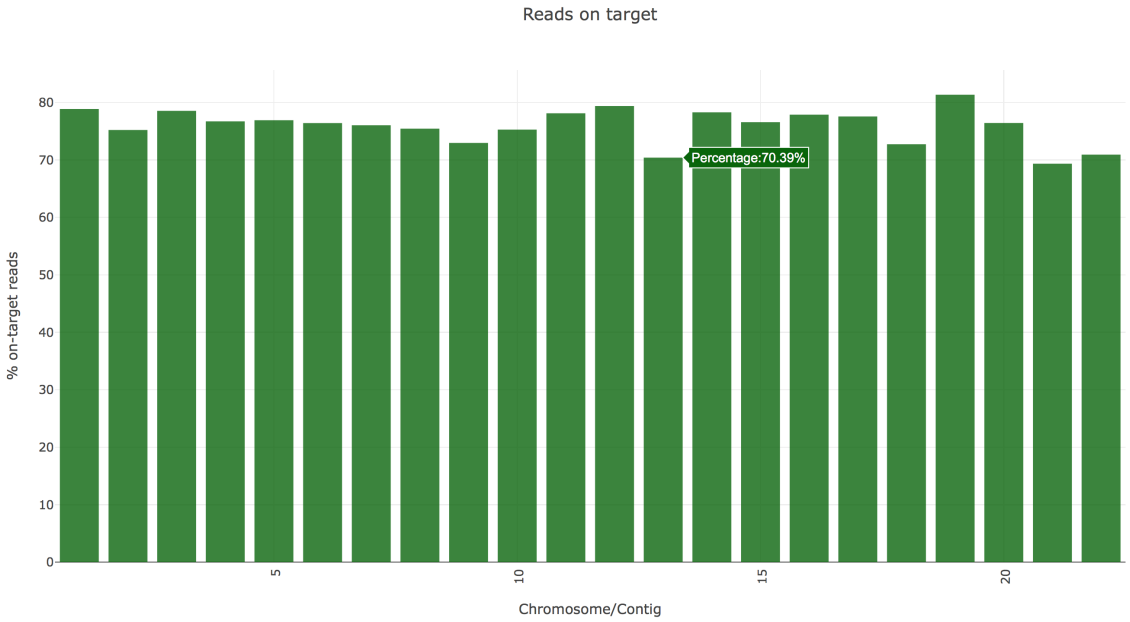

Figura 4.3: Gráfica del numero de lecturas en las regiones de interés. En ella están representada la proporción de las lecturas que alinean dentro de la región de interés y el total de las que alinean en el cromosoma.

Por otro lado en la segunda parte se realiza una cuantificación de lecturas duplicadas. Gracias a ella, como ya hemos comentado en la sección 2.3.1, podemos cuantificar el número de lecturas únicas y las repetidas (que son las que alinean exactamente en la misma localización cromosómica). En la figura 4.4 podemos ver representado con las barras azules las lecturas que alinean dentro de las regiones de interés y en color naranja las que se encuentran fuera de las regiones de interés. En este caso tenemos una mayor proporción de lecturas únicas en zonas fuera de las regiones de interés, indicándonos que no ha habido mucha amplificación de esas zonas. En cambio en las zonas dentro de nuestras ROIs vemos un aumento de las duplicidades, evento común en las zonas amplificadas, lo cuál nos indica que hay más regiones amplificadas dentro de nuestras regiones de interés que es el objetivo del paso del enriquecimiento. Como vemos, la interactividad en las gráficas introducida en esta nueva versión de la herramienta permite indicar cuales son los valores exactos tomados para la representación proporcionando de esta forma información adicional al usuario. Además se genera una tabla en formato XLS llamada *duplicates.xls*.

Finalmente se genera un archivo JSON que contiene la información de ambas métricas llamado *reads_on_target.json* del cual podemos ver un fragmento en la figura 4.5 y la totalidad de este en Anexo A.4. En él está representada la información detalla de los resultados generados por las métricas como por ejemplo, el número de lecturas totales que alinean en cada uno de los cromosomas, el número de lecturas duplicadas dentro de las ROIs y los umbrales utilizados en el experimento. Esta gráfica es configurable mediante la variable *maxduplicates* presente el archivo de configuración *config.json*

Por otro lado se generan archivos con formato BED de cada uno de los cromosomas con la

extensión *.off.annotated.bed* los cuales contienen las regiones fuera de los ROIs que presentan alta profundidad. Por defecto son aquellas regiones que están a más de 1000 bases de una región de interés y que tienen más 15x de profundidad. Ambos parámetros son modificables a través del archivo de configuración *config.json* mediante las variables *' offtargetoffset'* y *' offtargetthreshold'* respectivamente. Además, gracias a la nueva funcionalidad de anotación estos archivos se encuentran enriquecidos con la información que deseemos, en este caso con la información de los genes del genoma humano. En la figura 4.6 podemos ver un fragmento del archivo generado del cromosoma 18.

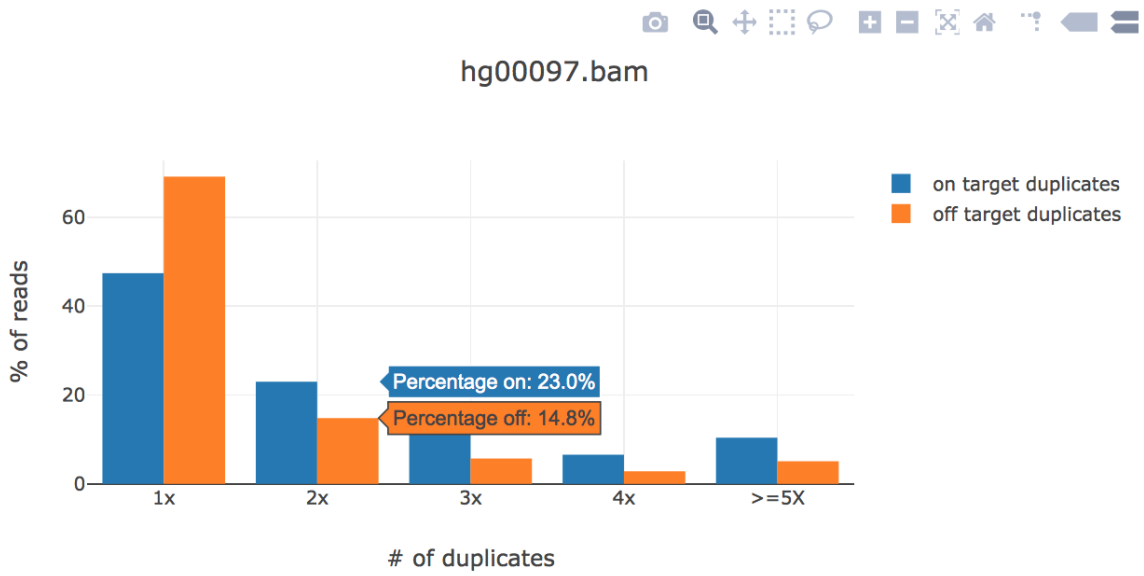

Figura 4.4: Porcentaje de lecturas duplicadas. En azul están representadas las lecturas dentro de las regiones de interés y en naranja fuera de estas. En este caso, el porcentaje de lecturas duplicadas tanto dentro como fuera de las regiones de interés que son del 23.0 % frente a 14.8 % respectivamente.


```
{
  "results": [
    {
      "bamfilename": "/home/agarcia/exomadata/hg00097.bam",
      "enrichment": 212.93288482175967,
      "onread": 43369690,
      "percontotal": 76.20651751894405,
      "totalread": 56910736,
      "onperchr": {
        "1": 4942773,
        "2": 3646987,
        "3": 2577714,
        "4": 1773878,
        "5": 2119788,
        "6": 1980103,
        "7": 2161669,
        "8": 1278642,
        "9": 1683661,
        "10": 1788881,
        "11": 2249571,
        "12": 2081953,
        "13": 707440,
        "14": 1299380,
        "15": 1489163,
        "16": 1739274,
        "17": 2341396,
        "18": 595955,
        "19": 2003756,
        "20": 890229,
        "21": 395000,
        "22": 857617,
        "X": 2760874,
        "Y": 3986
      },
      "totalperchr": {
        "1": 6268138,
        "2": 4849552,
```


Figura 4.5: Fragmento de read_on_target.json. En él encontramos toda la información de la especificidad.

```
18  14714548  14714549 19
18  14714549  14714552 18
18  14714552  14714553 17
18  14714553  14714559 16
18  14714559  14714560 15
18  14748394  14748397 15  NM_001145029_utr5_0_0_chr18_14748239_f
18  14748397  14748402 16  NM_001145029_utr5_0_0_chr18_14748239_f
18  14748402  14748404 17  NM_001145029_utr5_0_0_chr18_14748239_f
18  14748404  14748405 20  NM_001145029_utr5_0_0_chr18_14748239_f
18  14748405  14748411 19  NM_001145029_utr5_0_0_chr18_14748239_f
18  14748411  14748415 20  NM_001145029_utr5_0_0_chr18_14748239_f
18  14748415  14748416 21  NM_001145029_utr5_0_0_chr18_14748239_f
18  14748416  14748418 22  NM_001145029_utr5_0_0_chr18_14748239_f
18  14748418  14748419 23  NM_001145029_cds_0_0_chr18_14748419_f
18  14748419  14748424 24  NM_001145029_cds_0_0_chr18_14748419_f
18  14748424  14748425 25  NM_001145029_cds_0_0_chr18_14748419_f
18  14748425  14748426 26  NM_001145029_cds_0_0_chr18_14748419_f
```

Figura 4.6: Fragmento del archivo en formato BED con regiones fuera de los ROIs que presentan alta profundidad del cromosoma 18. De derecha a izquierda están representados, cromosoma, base de inicio, base de fin de la región, profundidad en la región, información proporcionada por la anotación

4.2.3. Uniformidad

La sección de uniformidad nos permite conocer cómo están distribuidas nuestras lecturas a lo largo de las regiones de interés que permiten conocer si hay sesgos debido a localizaciones cromosómicas o a composición nucleotídica. La uniformidad la componen. como vimos en la sección 2.3, cuatro métricas.

La primera métrica que encontramos en el informe final es la distribución de profundidad por posición (*coverage distribution*). En ella se generan dos gráficas. La primera de ellas es un histograma (Figura 4.7) donde podemos ver como la gran mayoría de las bases, en concreto 28,272,289, tienen de media 38 lecturas de profundidad. Recordar que este dato se ha visualizado únicamente situando el cursor encima de la barra. Por otro lado, se genera un gráfico de caja o *boxplot*, figura 4.8, donde podemos ver también la profundidad de las diferentes bases. Si situamos el cursor encima podemos ver los cuartiles, la media, la mediana, los máximos y mínimo. La muestra utilizada tiene una mediana de 59 lecturas de profundidad en cada bases.

Además de las gráficas se genera una tabla con la información de cuartiles, media, mediana, mínimos y máximos, llamada *percentile.xls* y un archivo en formato JSON con la información necesaria para representar el histograma por herramientas de visualización de terceros (*"hist-data"*), la información representada en la tabla y el número de bases dentro de las regiones de interés no cubiertas (*"zerocov"*).

Además como en todos los archivos de salida JSON que se generan está indicado el estado de la métrica (*"status"*) y el valor para ajustar el umbral *"warnthreshold"*. Por defecto, una señal de aviso salta si la media de las profundidades de las bases se encuentra por debajo del 40 %. Este valor es modificable mediante la variable *"warnmeancoverage"* en el archivo *config.json*.

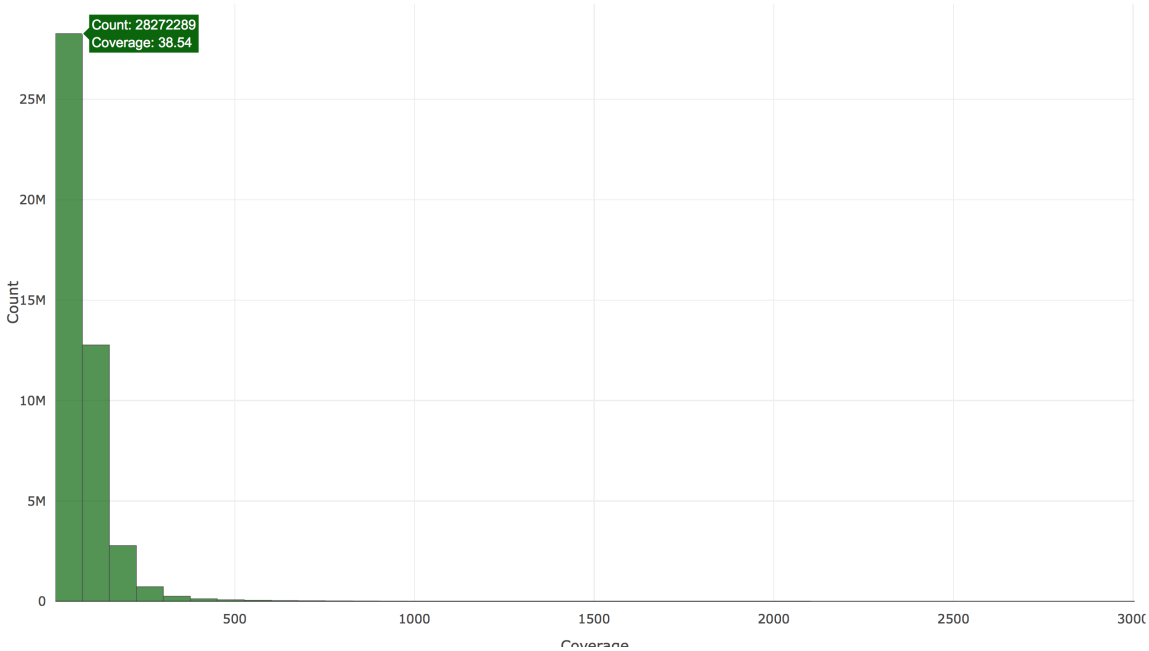

Figura 4.7: Histograma distribución de la profundidad por base. Al situar el cursor sobre cada una de las barras, aparece un desplegable con el número de bases y la profundidad media que tienen.

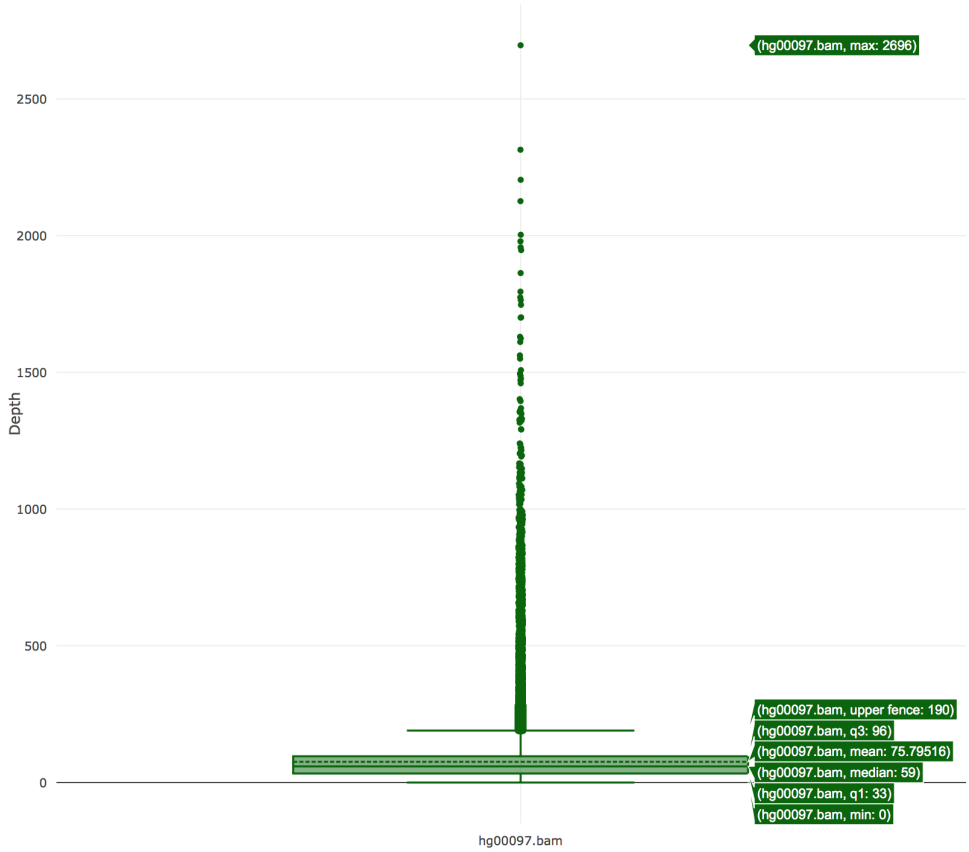

Figura 4.8: Gráfico de cajas distribución de la profundidad por base

Otra de las métricas que nos permite evaluar la uniformidad es la cobertura por posición (*coverage per position*) en la cual se representan las profundidades en las regiones de interés a

lo largo de cada uno de los cromosomas. Esta métrica nos permite ver si ha habido algún sesgo por localización, ya sea a nivel de cromosoma, o dentro de ellos. En el informe de *ngsCAT2* se encuentran representadas todas las gráficas de los cromosomas (Figura 4.9), para activar su interfaz interactiva es necesario hacer click en la gráfica de interés y automáticamente esta aumenta y permite la interactividad presente en la figura 4.10.

Como podemos ver en la figura 4.9, la mayoría de las regiones en cada uno de los cromosomas se encuentran cubiertas. Sin embargo, en el cromosoma Y (Figura 4.10) vemos que hay muchas regiones que no lo están. Cada uno de estos puntos de la gráfica representa a la media de las regiones englobadas, además en cada uno de ellos está representada la desviación estándar teniendo en cuenta todas las profundidades por base. Si el usuario desea obtener más información sobre estas regiones con poca cobertura únicamente debe de situar el cursor encima de alguno de los puntos y aparecerá una ventana con la información de cada una de las regiones que componen el punto seleccionado.

Para esta métrica en *ngsCAT2* se puede ajustar el número de puntos a representar por cromosoma lo cual, permite ajustar la resolución del los puntos a las necesidades de los usuarios. Esta configuración se puede realizar mediante la variable *npointsperchrom* (por defecto 50 puntos por cromosoma) presente en el archivo de configuración *config.json*.

Adicionalmente se genera un archivo de texto plano llamado *Nocoverage.annotated.txt* en el cual están descritas las regiones que no están cubiertas. Como hemos comentado y gracias a la nueva funcionalidad, las regiones no cubiertas también son anotadas.

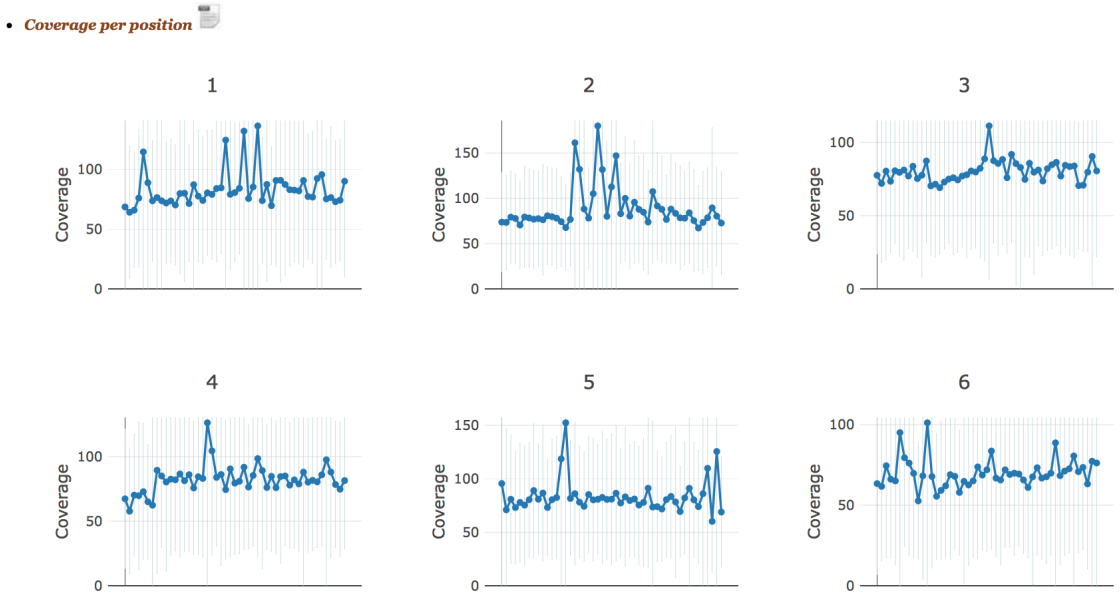

Figura 4.9: Gráficas de distribución de la profundidad a lo largo de las regiones de interés en cada uno de los cromosomas

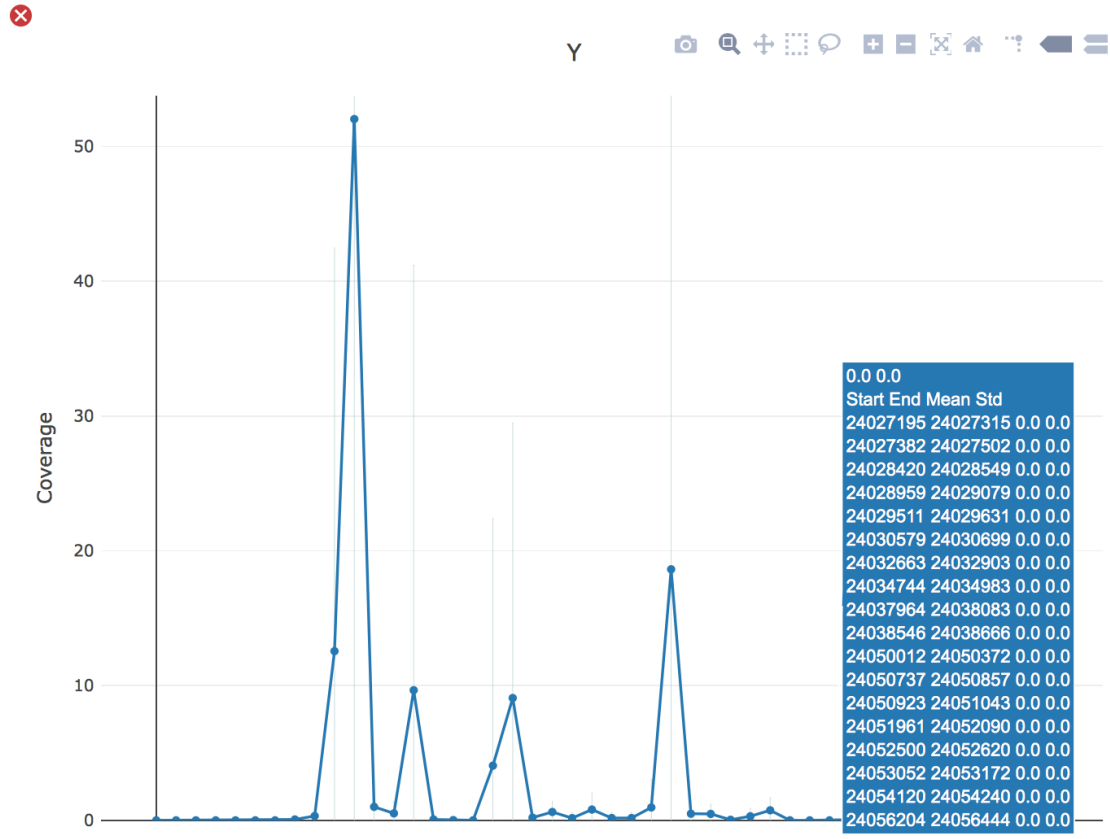

Figura 4.10: Cobertura y profundidad cromosoma Y. La baja cobertura presente este cromosoma probablemente se deba a que la muestra corresponda a una mujer (falta información del cromosoma Y).

```
1   15905   15910   NR_024540_utr3_2_0_chr1_15796_r
1   16600   16719   NR_024540_intron_2_0_chr1_15948_r;NR_024540_utr3_3_0_chr1_16607_r
1   129134  129253
1   228234  228279
1   228547  228711
1   334099  334218
1   367648  367764  NM_001005221_cds_0_0_chr1_367659_f;NM_001005224_cds_0_0_chr1_367659_f;NM_001005277_cds_0_0_chr1_367659_f
1   367866  367871  NM_001005221_cds_0_0_chr1_367659_f;NM_001005224_cds_0_0_chr1_367659_f;NM_001005277_cds_0_0_chr1_367659_f
1   368107  368608  NM_001005221_cds_0_0_chr1_367659_f;NM_001005224_cds_0_0_chr1_367659_f;NM_001005277_cds_0_0_chr1_367659_f
1   470972  471164
1   471267  471330
1   621085  621499  NM_001005221_cds_0_0_chr1_621096_r;NM_001005224_cds_0_0_chr1_621096_r;NM_001005277_cds_0_0_chr1_621096_r
1   621917  622045  NM_001005221_cds_0_0_chr1_621096_r;NM_001005224_cds_0_0_chr1_621096_r;NM_001005277_cds_0_0_chr1_621096_r
1   655491  655610
1   877761  877880  NM_152486_cds_9_0_chr1_877790_f;NM_152486_intron_8_0_chr1_877632_f;NM_152486_intron_9_0_chr1_877869_f
1   877907  878154  NM_152486_cds_10_0_chr1_877939_f;NM_152486_intron_9_0_chr1_877869_f
1   896008  896180  NM_198317_cds_0_0_chr1_896074_f;NM_198317_utr5_0_0_chr1_895967_f
1   896635  896661  NM_198317_intron_0_0_chr1_896181_f
```

Figura 4.11: Fragmento fichero de regiones no cubiertas anotado (NoCoverage). En el encontramos cromosoma, localización de inicio, localización de fin y los datos de la anotación. Gracias, a la anotación podemos buscar la región en la base de datos de RefSeq obteniendo más información sobre la región no cubierta. Como ejemplo hemos tomado la séptima linea anotación de la séptima linea del ejemplo NM_001005221, esta se corresponde con el mRNA que codifica a un miempro de los receptores olfativos (OR4F29)

La tercera métrica que permite evaluar la uniformidad de la profundidad es la desviación estándar de la cobertura en las regiones (*Standard deviation of the coverage within regions*). La desviación estándar normalizada de las profundidades de cada región es calculada mediante (desviación estándar / media de la profundidad) de cada una de las regiones de interés para posteriormente, generar un histograma y un gráfico de cajas. El histograma podemos verlo en la figura 4.12, en ella si situamos el cursor encima de una de las barras nos indica el número de

regiones representadas y el rango de la desviación estándar normalizada en el que se encuentran. Por otro lado las desviaciones también se representan en un gráfico de cajas 4.13, el cuál se encuentra en escala logarítmica, en ella si situamos el cursor encima podemos ver los cuartiles, la mediana, los máximos y mínimos. En este caso podemos ver que la desviación estándar de las profundidades en la mayoría de las regiones es mayor a 0.3. Esto nos indica que dentro de cada una de estas regiones de interés las lecturas que han alineado no están distribuidas uniformemente. En experimentos de secuenciación es deseable que las lecturas se encuentren alineadas de una manera más uniforme a lo largo de cada una de las regiones de interés.

Adicionalmente a la figuras, se genera una tabla en formato XLS llamado *std_wexons.xls* con los cuartiles, medias, máximos y mínimos. En esta versión además de los datos anteriores, en el archivo *std_wexons.json*, también se encuentran los datos necesarios para generar las gráficas y si se ha activado la señal de aviso en el informe. En esta métrica se puede modificar el umbral en que salta el aviso mediante la variable *warnstd*, que por defecto es 0.30, presente en el archivo de configuraciones *config.json*.

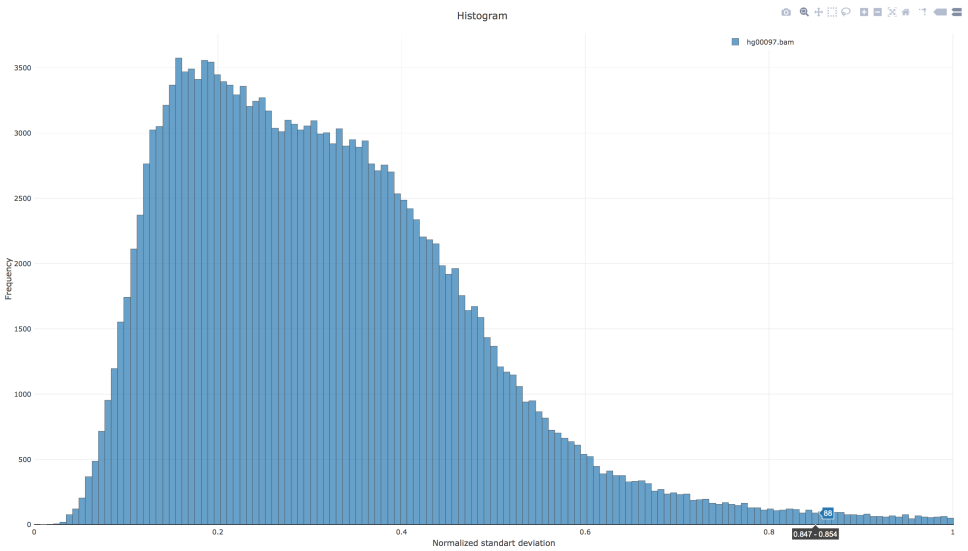

Figura 4.12: Histograma distribución de las desviaciones estándar normalizadas de las profundidades de las regiones de interés cada una de ellas calculadas mediante la siguiente fórmula (desviación estándar/ media profundidad).)

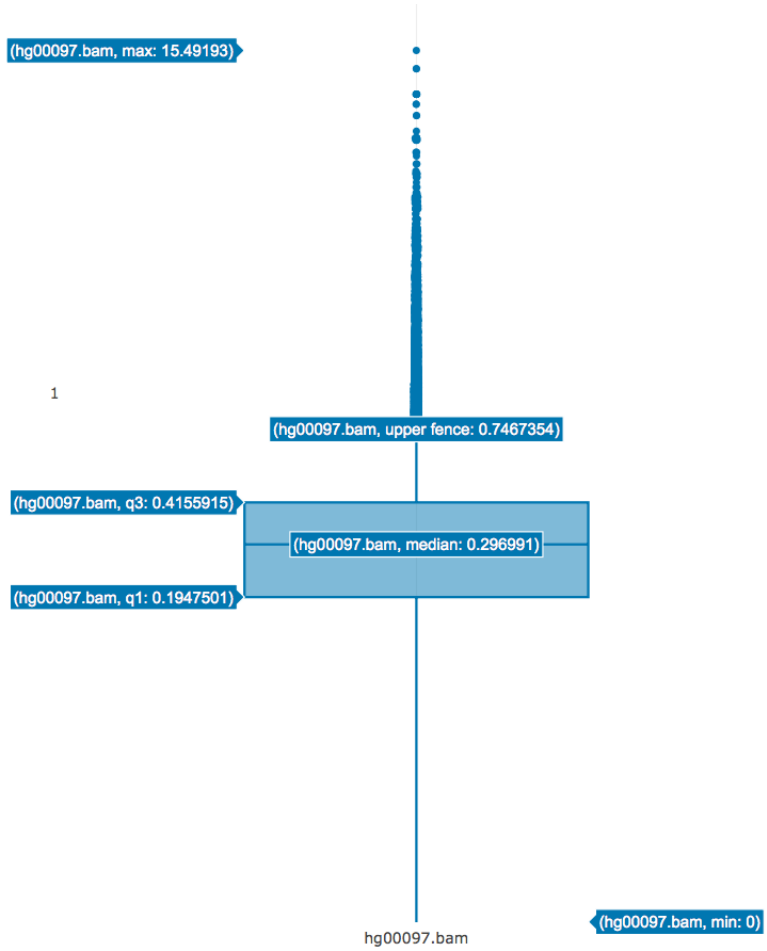

Figura 4.13: Gráfico de cajas de las desviaciones estándar normalizadas de la profundidad por región

La última métrica dentro del apartado de uniformidad es *GC Bias* o sesgo en contenido GC. En ella es generada la figura 4.14 que permite conocer la distribución de las profundidades en función del contenido en GC. La intensidad del color indica la densidad en esa zona, así pues mientras más intenso es el color mayor probabilidad de encontrar regiones de regiones de interés con la profundidad y proporción de contenido en GC. En la figura(4.14) podemos ver que en la muestra la profundidad es bastante uniforme independientemente del contenido en GC de las regiones, ya que no se observa una bajada del la profundidad en regiones con mayor contenido de GC.

Esta métrica es opcional, para su funcionamiento es necesario indicar en la llamada de *ngsCAT2*, mediante el argumento *–reference* un archivo en formato FASTA el cuál contendrá la composición nucleotídica.

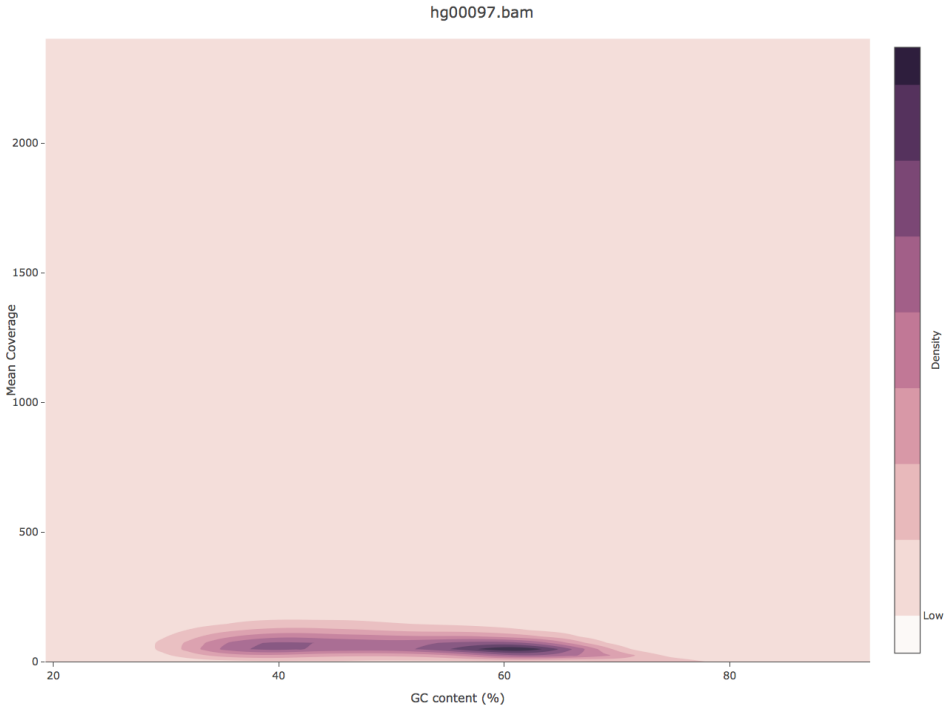

Figura 4.14: Gráfica de contenido en GC. En ella está representada la distribución de las medias de profundidad de las regiones frente al contenido en GC que tienen cada una de ellas

4.3. Comparativa de Rendimiento

Finalmente, se ha realizado una prueba comparativa de rendimiento entre la nueva herramienta *ngsCAT2* y *ngsCAT* en términos de tiempo de cómputo y de consumo de memoria. Las ejecuciones se han realizado con el exoma *hg0097* (descrito anteriormente) y únicamente utilizando las métricas por defecto. Para la medición del tiempo de ejecución se ha utilizado la función *time* de *linux* desde que se lanza la herramienta hasta que finaliza. Por otro lado, se ha realizado una medición aproximada del consumo de memoria de cada una de las herramientas. Para ello, se ha recurrido a la medición cada 5 segundos de la memoria libre del sistema. La razón por la cual se realiza una medición de la memoria libre es la dificultad de medir la memoria utilizada por cada uno de los programas debido a la paralelización de las métricas. Con todo esto, restando la memoria libre inicial (o lo que es lo mismo la memoria utilizada por el sistema operativo para funcionar) a las mediciones de memoria libre podremos conocer cual es el uso de memoria de las herramientas de una forma aproximada. Para obtener valores promedios, cada herramienta se ha ejecutado 10 veces.

En la tabla 4.1 podemos apreciar los tiempos de ejecución de las dos herramientas. Como podemos ver el tiempo de ejecución de la nueva herramienta aumenta ligeramente, no empleando significativamente mucho más tiempo que su predecesora. Este aumento de tiempo puede ser debido a la generación de todas las gráficas interactivas con formato HTML y en cierta medida al propio proceso de refactorización, aunque requiere de un estudio en mayor profundidad para determinar las causas de este aumento.

Cuadro 4.1: Tabla de tiempos de ejecución en ambas herramientas. En ella están representados los tiempos de ejecución medios tras las diez ejecuciones de ambas herramientas junto con las desviaciones estándar

Herramienta	Tiempo medio(min)
ngsCAT	$33{,}90 \pm 0{,}239$
ngsCAT2	$37{,}69 \pm 0{,}691$

Por otro lado, podemos ver los resultados del consumo de memoria en la tabla 4.2. Como era de esperar el consumo medio de RAM es mayor en la nueva versión, puesto que toda la información de profundidad es cargada en memoria. Sin embargo, el consumo de memoria máximo, que es el realmente limitante, vemos que en *ngsCAT2* es de 4602 Mb frente a 2905 Mb de *ngsCAT*. Esto no es un aumento desmesurado, teniendo en cuenta que en *ngsCAT2* se almacenan todos los datos de las profundidades en la memoria RAM. Tal y como se ha comentado en la parte de diseño e implementación (sección 3.3) el tamaño de los exomas está determinado y por tanto el volumen de los datos de profundidad almacenados en la memoria RAM se encuentra saturado.

Cuadro 4.2: Tabla de consumo de RAM en las 10 ejecuciones de ambas herramientas. Se encuentran reflejados los consumos de memoria medios y máximos junto con sus respectivas desviaciones estándar.

Herramienta	Uso de memoria medio(Mb)	Uso de memoria máximo(Mb)
ngsCAT	$485{,}75 \pm 318{,}45$	$2905{,}10 \pm 449{,}02$
ngsCAT2	$1308{,}90 \pm 1082{,}67$	$4602{,}90 \pm 163{,}07$

Como podemos ver, el tiempo de ejecución es ligeramente superior y consumo de memoria ha aumentado en esta nueva versión de la herramienta. Este descenso en el rendimiento no supone una gran desventaja en comparación con los beneficios obtenidos. En *ngsCAT2* se ha conseguido, gracias a la refactorización, una estructura más modularizada y ordenada. Además se ha simplificado la gestión de la concurrencia y se han añadido gráficas interactivas. Esto ha hecho que la nueva versión sea una herramienta mucho más flexible, donde es posible generar con la información proporcionada por las métricas nuevos informes y la inclusión de nuevas funcionalidades de una manera sencilla. Todo esto abre las puertas a nuevos desarrolladores y promueve el uso continuado de la herramienta en el tiempo.

5

Conclusiones y trabajo futuro

5.1. Conclusiones

En este trabajo se ha desarrollado una versión mejorada de la herramienta original *ngsCAT* que suple las carencias que presenta principalmente en su arquitectura, en la tecnología usada, en la instalación y en la representación de los datos. A continuación resumimos las mejoras introducidas y las conclusiones derivadas de la realización de este trabajo.

- Todo el proceso de refactorización y la aplicación de un nuevo diseño realizado en *ngsCAT2*, como la separación de las métricas y generadores de informe, la creación de una nueva estructura jerarquizada de generadores de informes, el cambio a un sistema de concurrencia basado en *pooling* y el nuevo sistema de almacenamiento en memoria de los datos, ha permitido crear una herramienta modular y flexible. De esta forma, la incorporación de nuevas métricas e informes por parte de desarrolladores externos es más sencilla.
- Gracias a la generación de las salidas de las métricas en formato JSON se ha conseguido aumentar la interoperabilidad, que permitirá la inclusión de los resultados generados por *ngsCAT2* en otras herramientas de terceros de control de calidad.
- Se ha incluido la anotación en los archivos de regiones de salida (formato BED) utilizando la dependencia *BEDOPS* aportando información biológica relevante a los usuarios de la herramienta.
- La inclusión de gráficas interactivas haciendo uso de el paquete *Plotly* ha enriquecido notablemente el informe generado, obteniendo así un informe más completo que permite ver y comparar mejor los resultados de cada una de las métricas.
- La instalación de la herramienta y de todas las dependencias de *Python* es ahora mucho más sencilla y rápida, gracias a la creación de un paquete de *Python* y su inclusión en la plataforma GitHub.
- La actualización de la tecnología usada, el paso a *Python* 3 y el uso de dependencias *Python* recientes, garantiza la continuidad y elminina problemas futuros de incompatibilidad, ya sea con otros programas o con las propias dependencias.

Todas estas mejoras han permitido obtener una herramienta mejorada de *ngsCAT*, mucho más completa, flexible y sencilla de desarrollar. De esta forma, la herramienta no solo puede ser usada por usuarios no expertos que deseen realizar estudios de evaluación de calidad en experimentos de secuenciación dirigida, sino también por desarrolladores que deseen mejorarla gracias a su nueva arquitectura software, permitiendo un aumento de su soporte y permanencia en la comunidad científica.

5.2. Trabajo Futuro

Aunque como ya hemos indicado, se ha obtenido una herramienta mejorada de *ngsCAT*, existen ciertos aspectos que pueden ser abordados en el futuro con el objetivo de seguir mejorando la herramienta.

- El uso de una plantilla HTML para generar el informe final en este formato puede frenar a los futuros desarrolladores de la aplicación, ya que si el usuario necesita añadir nuevas funciones debe de cambiar o crear nuevas plantillas. Además, en la nueva versión mejorada se hace uso de funciones como *iframe* para insertar las gráficas interactivas, lo cual puede generar problemas de visualización en algunos navegadores web. Por lo tanto, es deseable utilizar nuevos métodos para la creación del informe final, como puede ser una inyección directa del código HTML de los resultados y de las gráficas conformando directamente el HTML, eliminando así el uso de plantillas, que entorpece la adicción de secciones nuevas en el informe, y la utilización de funciones del lenguaje HTML que pueden generar problemas de visualización.
- Previo a la refactorización de *ngsCAT* se han realizado pruebas de velocidad utilizando diferentes librerías de la lectura de los archivos de alineamientos (BAM). Sin embargo, ninguna de las utilizadas permitía la paralelización de la lectura de estos. Por ello, es necesaria mayor investigación de librerías que tengan esta opción o diseñar nuevas funciones que permitan la paralelización de la lectura de los pesados archivos de alineamiento. El uso de software que permita la lectura de los archivos BAM de una manera concurrente, disminuiría considerablemente la velocidad del análisis ya que la lectura consume gran parte del tiempo total de ejecución de la herramienta.
- La creación del paquete de *Python* favorece enormemente la instalación de la aplicación y los módulos de los que depende. Sin embargo, *ngsCAT2*, al igual que *ngsCAT*, hace uso de dependencias externas. La instalación de estas dependencias como *BEDOPS* o *Samtools* dificulta en cierto modo el uso de la herramienta por personas no familiarizadas con la terminal de *linux*. Actualmente, existen proyectos como *Docker* que permite generar imágenes virtuales ligeras y portables para las aplicaciones y sus dependencias. Gracias a *Docker* estas imágenes pueden ser ejecutadas en cualquier maquina, independientemente del sistema operativo. Además, elimina la necesidad de instalar dependencias puesto que ya están incluidas. Así pues, como posible trabajo futuro, se plantea la utilización de *Docker* para crear una imagen con el paquete *ngsCAT2*, *Samtools* y *BEDOPS*, para con ello, facilitar más aun la instalación y ejecución de la herramienta.
- Tras el cambio de arquitectura ha disminuido ligeramente el rendimiento de la herramienta, por ello es necesario testear el programa de una manera aislada, permitiendo así identificar posibles cuellos de botella, para posteriormente intentar optimizar el consumo de recursos.

Glosario de acrónimos

- **ROI**: Region of interest (Región de interés)
- **NGS**: Next Generation Sequencing
- **WES**: Whole Exome Sequencing (Secuenciación de exoma completo), comprende todas las regiones del genoma que codifican proteínas
- **DNA**: Deoxyribonucleic acid (Ácido desoxirribonucleico)
- **RNA**: Ribonucleic acid (Ácido ribonucleico)
- **BAM**: Binary alignment/Map (Binario de alineamiento mapeo) (32)
- **BED**: Browser Extensible Data, https://www.ensembl.org/info/website/upload/bed.html
- **CPU**: Central Processing Unit (Unidad central de Procesamiento)
- **RAM**: Random-Access Memory (Memoria de acceso aleatorio)
- **JSON**:JavaScript Object Notation, notación de objeto de javascript
- **XLS**: eXceL Spreadsheet (Formato de tabla creado por *Microsoft*)
- **PNG**: Portable network graphic
- **FASTA**: Formato basado en texto plano para representar tanto secuencias nucleotídicas como secuencias peptídicas.

Bibliografía

[1] K. Wetterstrand, "DNA Sequencing Costs: Data from the NHGRI Genome Sequencing Program (GSP) Available at: www. genome. gov/sequencingcosts. Accessed." www.genome.gov/sequencingcostsdata, 2014.

[2] E. A. Normand, J. T. Alaimo, and I. B. Van den Veyver, "Exome and genome sequencing in reproductive medicine," *Fertility and Sterility*, vol. 109, no. 2, pp. 213–220, 2018.

[3] D. F. Easton, P. D. Pharoah, A. C. Antoniou, M. Tischkowitz, S. V. Tavtigian, K. L. Nathanson, P. Devilee, A. Meindl, F. J. Couch, M. Southey, D. E. Goldgar, D. G. R. Evans, G. Chenevix-Trench, N. Rahman, M. Robson, S. M. Domchek, and W. D. Foulkes, "Gene-Panel Sequencing and the Prediction of Breast-Cancer Risk," *New England Journal of Medicine*, vol. 372, no. 23, pp. 2243–2257, 2015.

[4] T. Trusight and O. Series, "TruSight ^{TM} One Series of Sequencing Panels," pp. 1–4, 2013.

[5] S. L. Sawyer, T. Hartley, D. A. Dyment, C. L. Beaulieu, J. Schwartzentruber, A. Smith, H. M. Bedford, G. Bernard, F. P. Bernier, B. Brais, D. E. Bulman, J. Warman Chardon, D. Chitayat, J. Deladoëy, B. A. Fernandez, P. Frosk, M. T. Geraghty, B. Gerull, W. Gibson, R. M. Gow, G. E. Graham, J. S. Green, E. Heon, G. Horvath, A. M. Innes, N. Jabado, R. H. Kim, R. K. Koenekoop, A. Khan, O. J. Lehmann, R. Mendoza-Londono, J. L. Michaud, S. M. Nikkel, L. S. Penney, C. Polychronakos, J. Richer, G. A. Rouleau, M. E. Samuels, V. M. Siu, O. Suchowersky, M. A. Tarnopolsky, G. Yoon, F. R. Zahir, J. Majewski, and K. M. Boycott, "Utility of whole-exome sequencing for those near the end of the diagnostic odyssey: Time to address gaps in care," *Clinical Genetics*, vol. 89, no. 3, pp. 275–284, 2016.

[6] E. M. Miller, N. E. Patterson, J. M. Zechmeister, M. Bejerano-Sagie, M. Delio, K. Patel, N. Ravi, W. Quispe-Tintaya, A. Maslov, N. Simmons, M. Castaldi, J. Vijg, R. G. Karabakhtsian, J. M. Greally, D. Y. Kuo, and C. Montagna, "Development and validation of a targeted next generation DNA sequencing panel outperforming whole exome sequencing for the identification of clinically relevant genetic variants," *Oncotarget*, vol. 8, no. 60, pp. 102033–102045, 2017.

[7] R. Tewhey, M. Nakano, X. Wang, C. Pabón-Peña, B. Novak, A. Giuffre, E. Lin, S. Happe, D. N. Roberts, E. M. LeProust, E. J. Topol, O. Harismendy, and K. A. Frazer, "Enrichment of sequencing targets from the human genome by solution hybridization," *Genome Biology*, vol. 10, no. 10, p. R116, 2009.

[8] Asan, Y. Xu, H. Jiang, C. Tyler-Smith, Y. Xue, T. Jiang, J. Wang, M. Wu, X. Liu, G. Tian, J. Wang, J. Wang, H. Yang, and X. Zhang, "Comprehensive comparison of three commercial human whole-exome capture platforms," *Genome Biology*, vol. 12, no. 9, p. R95, 2011.

[9] M. J. Clark, R. Chen, H. Y. Lam, K. J. Karczewski, R. Chen, G. Euskirchen, A. J. Butte, and M. Snyder, "Performance comparison of exome DNA sequencing technologies," *Nature Biotechnology*, vol. 29, no. 10, pp. 908–916, 2011.

[10] F. J. López-Domingo, J. P. Florido, A. Rueda, J. Dopazo, and J. Santoyo-Lopez, "NgsCAT: A tool to assess the efficiency of targeted enrichment sequencing," *Bioinformatics*, vol. 30, no. 12, pp. 1767–1768, 2014.

[11] J. Dopazo, A. Amadoz, M. Bleda, L. Garcia-Alonso, A. Alemán, F. García-García, J. A. Rodriguez, J. T. Daub, G. Muntané, A. Rueda, A. Vela-Boza, F. J. López-Domingo, J. P. Florido, P. Arce, M. Ruiz-Ferrer, C. Méndez-Vidal, T. E. Arnold, O. Spleiss, M. Alvarez-Tejado, A. Navarro, S. S. Bhattacharya, S. Borrego, J. Santoyo-López, and G. Antiñolo, "267 Spanish Exomes Reveal Population-Specific Differences in Disease-Related Genetic Variation," *Molecular Biology and Evolution*, vol. 33, no. 5, pp. 1205–1218, 2016.

[12] P. T. Inc., "Collaborative data science." https://plot.ly, 2015.

[13] A. M. Maxam and W. Gilbert, "A new method for sequencing DNA.," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 74, pp. 560–4, feb 1977.

[14] F. Sanger, S. Nicklen, and A. R. Coulson, "DNA sequencing with chain-terminating inhibitors.," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 74, no. 12, pp. 5463–7, 1977.

[15] E. D. Green, J. D. Watson, and F. S. Collins, "Human Genome Project: Twenty-five years of big biology," *Nature*, vol. 526, no. 7571, pp. 29–31, 2015.

[16] J. Shendure and H. Ji, "Next-generation DNA sequencing.," *Nature biotechnology*, vol. 26, no. 10, pp. 1135–45, 2008.

[17] C. Thermes, "Ten years of next-generation sequencing technology," *Trends in genetics : TIG*, vol. 30, no. 9, pp. 418–426, 2014.

[18] PacBio, "SMRT PacBio page." https://www.pacb.com/smrt-science/smrt-sequencing/, 2019.

[19] M. C. M. F. Michele Araújo Pereira, Frederico Scott Varella Malta and P. G. P. Couto, "Application of Next-Generation Sequencing in the Era of Precision Medicine," *InTech*, p. 43936, sep.

[20] L. Mamanova, A. J. Coffey, C. E. Scott, I. Kozarewa, E. H. Turner, A. Kumar, E. Howard, J. Shendure, and D. J. Turner, "Target-enrichment strategies for next-generation sequencing," *Nature Methods*, vol. 7, no. 2, pp. 111–118, 2010.

[21] Navarrabiomed, "Proyecto 1000 Genoma Navarra, NAGEN 1000." https://www.nagen1000navarra.es/es/incio, 2019.

[22] "The 100 000 Genomes Project: Bringing whole genome sequencing to the NHS," *BMJ (Online)*, vol. 361, p. k1952, may 2018.

[23] N. Mahdieh and B. Rabbani, "An overview of mutation detection methods in genetic disorders.," *Iranian journal of pediatrics*, vol. 23, no. 4, pp. 375–88, 2013.

[24] E. R. Mardis, "The $1,000 genome, the $100,000 analysis?," *Genome medicine*, vol. 2, no. 11, p. 84, 2010.

[25] A. R. Quinlan and I. M. Hall, "BEDTools: A flexible suite of utilities for comparing genomic features," *Bioinformatics*, vol. 26, no. 6, pp. 841–842, 2010.

[26] BroadInstitue, "Picard tools: MarkDuplicates." http://broadinstitute.github.io/picard/, 2018.

[27] P. Frommolt, A. T. Abdallah, J. Altmüller, S. Motameny, H. Thiele, C. Becker, K. Stemshorn, M. Fischer, T. Freilinger, and P. Nürnberg, "Assessing the Enrichment Performance in Targeted Resequencing Experiments," *Human Mutation*, vol. 33, no. 4, pp. 635–641, 2012.

[28] G. A. Merino, C. Fresno, Y. Murua, A. S. Llera, and E. A. Fernández, "TarSeqQC : Targeted Sequencing Experiment Quality Control," 2018.

[29] M. Hummel, S. Bonnin, E. Lowy, and G. Roma, "TEQC: An R package for quality control in target capture experiments," *Bioinformatics*, vol. 27, no. 9, pp. 1316–1317, 2011.

[30] M. Münz, S. Mahamdallie, S. Yost, A. Rimmer, E. Poyastro-Pearson, A. Strydom, S. Seal, E. Ruark, and N. Rahman, "CoverView: a sequence quality evaluation tool for next generation sequencing data," *Wellcome Open Research*, vol. 3, no. May, p. 36, 2018.

[31] "Python Flask Framework." http://flask.pocoo.org/.

[32] H. Li, B. Handsaker, A. Wysoker, T. Fennell, J. Ruan, N. Homer, G. Marth, G. Abecasis, and R. Durbin, "The Sequence Alignment/Map format and SAMtools," *Bioinformatics*, vol. 25, no. 16, pp. 2078–2079, 2009.

[33] M. Fowler, D. Martin Fowler, K. Beck, J. C. Shanklin, Addison-Wesley, E. Gamma, S. T. B. O. O. Service), J. Brant, W. Opdyke, D. Roberts, and Others, *Refactoring: Improving the Design of Existing Code*. Addison-Wesley object technology series, Addison-Wesley, 1999.

[34] S. Neph, M. S. Kuehn, A. P. Reynolds, E. Haugen, R. E. Thurman, A. K. Johnson, E. Rynes, M. T. Maurano, J. Vierstra, S. Thomas, R. Sandstrom, R. Humbert, and J. A. Stamatoyannopoulos, "BEDOPS: High-performance genomic feature operations," *Bioinformatics*, vol. 28, pp. 1919–1920, jul 2012.

[35] P. Ewels, M. Magnusson, S. Lundin, and M. Käller, "MultiQC: summarize analysis results for multiple tools and samples in a single report," *Bioinformatics (Oxford, England)*, vol. 32, pp. 3047–3048, oct 2016.

A

Manual de utilización

A.1. Instalación de la herramienta

Primero de todo comenzaremos con la instalación de la herramienta *ngsCAT2*. Para ello podemos seguir las instrucciones presentes en el documento de ayuda de la herramienta en GitHub https://github.com/babelomics/ngscat2/blob/master/README.md. Antes de esto es necesario tener instalado en la máquina *Python* 3 y el gestor de paquetes de paquetes *pip3*. También es necesario tener instalados los programas *Samtools* y *BEDOPS*. Una vez realizado esto, para la instalación de *ngsCAT2* únicamente es necesario ejecutar una linea en el terminal, y automáticamente descargará la última versión subida a GitHub e instalará todas las dependencias *Python*.

```
1 pip3 install git+https://github.com/babelomics/ngscat2@master
```

Caja de código A.1: Comando shell para la instalacíon de *ngsCAT2*

A.2. Manual de uso

ón de ayuda de la herramienta ngsCAT2

```
Usage:

    ********************************************************************************

        Task: Assesses capture performance in terms of sensibility, specificity
    and uniformity of the coverage.
        Output: An html report will be created at the path indicated with the --
    out option.

    ********************************************************************************

        usage: ngscat2 --bams <filename> --bed <filename> --out <path> --
    annotation <filename> --reference <filename>  --tmp <path> --threads <integer>

Options:
  -h, --help            show this help message and exit
  --bams=BAMS           Required. Comma separated list of bam files (2
                        maximum). E.g.: --bams
                        /home/user/bam1.sorted.bam,/home/user/bam2.sorted.bam
  --bed=BED             Required. Full path to the bed file containing the
                        target regions.
  --out=OUTDIR          Required. Full path to the directory where results
                        will be saved.
  --reference=REFERENCE
                        Optional. String indicating the path to a .fasta file
                        containing the reference chromosomes. Default=None.
  --annotation=ANNOTATION
                        Optional. String indicating the path to a .bed file
                        containing annotated regions . Default=None.
  --coveragethrs=COVERAGETHRESHOLDS
                        Optional. Comma separated list of real numbers (do not
                        leave spaces between) indicating coverage thresholds
                        to be used when calculating percentages of covered
                        bases (first graph in the report).
                        Default=1,5,10,20,30.
  --tmp=TMP             Optional. String indicating the full path to a
                        temporary directory where temporary files will be
                        created. Default=/tmp/.
  --threads=NTHREADS    Optional. Integer indicating the number of concurrent
                        threads to launch. Default=cpu_count() - 1.
```

A.3. Scripts usados para la prueba comparativa

```
#!/bin/bash
for i in {1..10}
    do
        ./memuse.sh ngscat2_mem_$i.txt &
        WATCHPID=$!
        /usr/bin/time -f'%e' -o  ngscat2_time.txt -a  ngscat2 --bam /home/agarcia
    /exomadata/hg00097.bam --bed /home/agarcia/exomadata/exome.targets.g1k.bed --
    out /home/agarcia/exomeresults
        kill $WATCHPID
        rm -r /home/agarcia/exomeresults
    done
```

Caja de código A.2: Script medición tiempo y memoria utilizado. Timeloop.sh

```
#!/bin/bash
while true
do
    free -m|grep Mem >> $1
    sleep 5
done
```

Caja de código A.3: Script memuse.sh

A.4. Salidas generadas por ngsCAT2


```
{
  "results": [
    {
      "bamfilename": "/home/agarcia/exomadata/hg00097.bam",
      "enrichment": 212.93288482175967,
      "onread": 43369690,
      "percontotal": 76.20651751894405,
      "totalread": 56910736,
      "onperchr": {
        "1": 4942773,
        "2": 3646987,
        "3": 2577714,
        "4": 1773878,
        "5": 2119788,
        "6": 1980103,
        "7": 2161669,
        "8": 1278642,
        "9": 1683661,
        "10": 1788881,
        "11": 2249571,
        "12": 2081953,
        "13": 707440,
        "14": 1299380,
        "15": 1489163,
        "16": 1739274,
        "17": 2341396,
        "18": 595955,
        "19": 2003756,
        "20": 890229,
        "21": 395000,
        "22": 857617,
        "X": 2760874,
        "Y": 3986
      },
      "totalperchr": {
        "1": 6268138,
        "2": 4849552,
        "3": 3282358,
        "4": 2312623,
        "5": 2756477,
        "6": 2591558,
        "7": 2843266,
        "8": 1695042,
        "9": 2307904,
        "10": 2376659,
        "11": 2880097,
        "12": 2623019,
        "13": 1004969,
        "14": 1659992,
        "15": 1945045,
        "16": 2233592,
        "17": 3018994,
        "18": 819470,
        "19": 2463026,
        "20": 1164912,
        "21": 569680,
        "22": 1209336,
        "X": 3706318,
        "Y": 5626
      },
      "perconperchr": {
        "1": 78.85552296391688,
        "2": 75.20255479269012,
        "3": 78.53238434076965,
        "4": 76.70415800586606,
        "5": 76.90207464092754,
        "6": 76.40589174542882,
        "7": 76.02767380892256,
        "8": 75.43423702775506,
        "9": 72.95195120767588,
        "10": 75.26872807584091,
        "11": 78.10747346356737,
        "12": 79.37239493880907,
        "13": 70.39421116472249,
        "14": 78.27628084954627,
        "15": 76.5618790310764,
        "16": 77.86892145029172,
        "17": 77.5555035882814,
        "18": 72.72444384785288,
        "19": 81.35342460859123,
        "20": 76.42027895669372,
        "21": 69.33717174554135,
        "22": 70.91635409844741,
        "X": 74.49101777019673,
        "Y": 70.84962673302525
      },
      "retonduplicates": [
        40.06156764516277
      ],
      "retoffduplicates": [
        6.768552422165126
      ],
      "legend": "hg00097.bam",
      "onduplicates": {
        "1x": 20570357,
        "2x": 9976028,
```

```
        "3x": 5462604,
        "4x": 2849516,
        "5x": 1515550,
        "6x": 858618,
        "7x": 528423,
        "8x": 351848,
        "9x": 252450,
        "10x": 188940,
        "11x": 147202,
        "12x": 117000,
        "13x": 94237,
        "14x": 78386,
        "15x": 63555,
        "16x": 53536,
        "17x": 44251,
        "18x": 35748,
        "19x": 29241,
        "20x": 24560,
        "21x": 20307,
        "22x": 17578,
        "23x": 13938,
        "24x": 12144,
        "25x": 9000,
        "26x": 8710,
        "27x": 7236,
        "28x": 5488,
        "29x": 5742,
        "30x": 4470,
        "31x": 3968,
        "32x": 3264,
        "33x": 2673,
        "34x": 2176,
        "35x": 1435,
        "36x": 1224,
        "37x": 1406,
        "38x": 1140,
        "39x": 936,
        "40x": 680,
        "41x": 615,
        "42x": 546,
        "43x": 172,
        "44x": 132,
        "45x": 405,
        "46x": 230,
        "47x": 188,
        "48x": 288,
        "49x": 147,
        "50x": 300,
        "51x": 153,
        "52x": 0,
        "53x": 53,
        "54x": 0,
        "55x": 55,
        "56x": 168,
        "57x": 114,
        "58x": 174,
        "59x": 59,
        "60x": 0,
        "61x": 61,
        "62x": 0,
        "63x": 63,
        "64x": 0,
        "65x": 0,
        "66x": 66,
        "67x": 0,
        "68x": 136
      },
      "offduplicates": {
        "1x": 9365930,
        "2x": 2004586,
        "3x": 773730,
        "4x": 384220,
        "5x": 211580,
        "6x": 127812,
        "7x": 85337,
        "8x": 58416,
        "9x": 42678,
        "10x": 32360,
        "11x": 25377,
        "12x": 19416,
        "13x": 14911,
        "14x": 12390,
        "15x": 9630,
        "16x": 8176,
        "17x": 6647,
        "18x": 5688,
        "19x": 4560,
        "20x": 3500,
        "21x": 3171,
        "22x": 2838,
        "23x": 2185,
        "24x": 2448,
        "25x": 1475,
        "26x": 1170,
        "27x": 1188,
        "28x": 924,
        "29x": 696,
        "30x": 630,
        "31x": 651,
```

```
        "32x": 672,
        "33x": 660,
        "34x": 544,
        "35x": 315,
        "36x": 216,
        "37x": 148,
        "38x": 342,
        "39x": 195,
        "40x": 40,
        "41x": 82,
        "42x": 42,
        "43x": 43,
        "44x": 0,
        "45x": 0,
        "46x": 46,
        "47x": 47,
        "48x": 48,
        "49x": 49,
        "50x": 50,
        "51x": 51,
        "52x": 0,
        "53x": 53,
        "54x": 0,
        "55x": 0,
        "56x": 0,
        "57x": 0,
        "58x": 0,
        "59x": 0,
        "60x": 0,
        "61x": 0,
        "62x": 0,
        "63x": 0,
        "64x": 0,
        "65x": 0,
        "66x": 0,
        "67x": 0,
        "68x": 0
      },
      "perconduplicates": {
        "1x": 47.4302606267188,
        "2x": 23.002304143746475,
        "3x": 12.595441655220501,
        "4x": 6.570293677450772,
        "5x": 3.4944911988072778,
        "6x": 1.9797651309013276,
        "7x": 1.2184154417520623,
        "8x": 0.8112762622928593,
        "9x": 0.5820885507828163,
        "10x": 0.4356498743707875,
        "11x": 0.33941215627780597,
        "12x": 0.26977365989934443,
        "13x": 0.217287695623372,
        "14x": 0.1807391291014531,
        "15x": 0.14654243551199006,
        "16x": 0.12344104834505387,
        "17x": 0.10203208738637515,
        "18x": 0.08242622900924586,
        "19x": 0.06742266315484385,
        "20x": 0.056629411001093165,
        "21x": 0.04682302317586314,
        "22x": 0.04053061020265536,
        "23x": 0.03213765189467575,
        "24x": 0.0280011224428858,
        "25x": 0.020751819992257264,
        "26x": 0.0200831502369512,
        "27x": 0.01668446327377484,
        "28x": 0.012653998679723099,
        "29x": 0.013239661155060135,
        "30x": 0.010306737262821108,
        "31x": 0.009149246858808536,
        "32x": 0.007525993383858635,
        "33x": 0.006163290537700408,
        "34x": 0.005017328922572423,
        "35x": 0.003308762409876575,
        "36x": 0.002822247518946988,
        "37x": 0.0032418954343459688,
        "38x": 0.002628563865685 92,
        "39x": 0.0021581892791947554,
        "40x": 0.0015679152883038823,
        "41x": 0.0014180410328042466,
        "42x": 0.0012589437461969408,
        "43x": 0.00039659033762980546,
        "44x": 0.00030436002655310655,
        "45x": 0.0009338318996515 77,
        "46x": 0.000530324288691019,
        "47x": 0.00043348246206048515,
        "48x": 0.0006640582397522324,
        "49x": 0.00033894639320686867,
        "50x": 0.0006917273330752422,
        "51x": 0.0003527809398683735,
        "52x": 0,
        "53x": 0.00012220516217662612,
        "54x": 0,
        "55x": 0.00012681667773046108,
        "56x": 0.0003873673065221356,
        "57x": 0.00026285638656859203,
        "58x": 0.00040120185318364045,
        "59x": 0.00013603970883813095,
        "60x": 0,
```

```
        "61x": 0.0001406512243919659,
        "62x": 0,
        "63x": 0.00014526273994580085,
        "64x": 0,
        "65x": 0,
        "66x": 0.00015218001327655327,
        "67x": 0,
        "68x": 0.0003135830576607764
      },
      "percoffduplicates": {
        "1x": 69.1669609570782,
        "2x": 14.803775129336389,
        "3x": 5.713960354318271,
        "4x": 2.8374469741850077,
        "5x": 1.562508538852907,
        "6x": 0.943885723451497,
        "7x": 0.6302098080163083,
        "8x": 0.4313994650044022,
        "9x": 0.315175061069876,
        "10x": 0.2389771070861143,
        "11x": 0.18740797424364408,
        "12x": 0.1433862642516686,
        "13x": 0.11011704708779513,
        "14x": 0.09149957839298382,
        "15x": 0.07111710572432883,
        "16x": 0.06037938280395768,
        "17x": 0.04908778834367744,
        "18x": 0.04200561758670637,
        "19x": 0.03367538962647346,
        "20x": 0.025847338529091473,
        "21x": 0.023417688707356877,
        "22x": 0.02095849907016046,
        "23x": 0.016136124196018533,
        "24x": 0.018078367062633122,
        "25x": 0.010892806951545693,
        "26x": 0.008640396022582007,
        "27x": 0.00877332519216019,
        "28x": 0.00682369737168015,
        "29x": 0.005139927890356476,
        "30x": 0.004652520935236465,
        "31x": 0.004807604966411015,
        "32x": 0.0049626889975855635,
        "33x": 0.004874069551200106,
        "34x": 0.0040174149028073605,
        "35x": 0.0023262604676182325,
        "36x": 0.0015951500349382166,
        "37x": 0.0010929731720872966,
        "38x": 0.00252565422198551,
        "39x": 0.0014400660037636678,
        "40x": 0.00029539815461818825,
        "41x": 0.000605566216967286,
        "42x": 0.0003101680623490977,
        "43x": 0.0003175530162145524,
        "44x": 0,
        "45x": 0,
        "46x": 0.00033970787781091653,
        "47x": 0.00034709283167637123,
        "48x": 0.00035447778554182594,
        "49x": 0.00036186273940728064,
        "50x": 0.00036924769327273535,
        "51x": 0.00037663264713819005,
        "52x": 0,
        "53x": 0.0003914025548690995,
        "54x": 0,
        "55x": 0,
        "56x": 0,
        "57x": 0,
        "58x": 0,
        "59x": 0,
        "60x": 0,
        "61x": 0,
        "62x": 0,
        "63x": 0,
        "64x": 0,
        "65x": 0,
        "66x": 0,
        "67x": 0,
        "68x": 0
      },
      "onoff_status": "warning",
      "duplicates_status": "ok",
      "warnthreshold": 80,
      "maxduplicates": 5
    }
  ],
  "bedfile": "/home/agarcia/exomadata/exome.targets.g1k.bed",
  "maxduplicates": 5
}
```


Caja de código A.4: Archivo en formato JSON reads_on_target.json

```
[
  {
    "bamfilename": "/home/agarcia/exomadata/hg00097.bam",
    "legend": "hg00097.bam",
    "histdata": {
      "numberread": [
        28272289,
        12767221,
        2777879,
        728416,
        256106,
        126783,
        78241,
        54059,
        38827,
        28991,
        21326,
        16523,
        12388,
        9683,
        7815,
        5846,
        4425,
        3949,
        2867,
        2074,
        1677,
        1271,
        1275,
        1069,
        660,
        621,
        454,
        435,
        243,
        179,
        158,
        191,
        147,
        187,
        74,
        45,
        22,
        36,
        14,
        17
      ],
      "binedges": [
        1,
        76.075,
        151.15,
        226.22500000000002,
        301.3,
        376.375,
        451.45000000000005,
        526.525,
        601.6,
        676.6750000000001,
        751.75,
        826.825,
        901.9000000000001,
        976.975,
        1052.05,
        1127.125,
        1202.2,
        1277.275,
        1352.3500000000001,
        1427.425,
        1502.5,
        1577.575,
        1652.65,
        1727.7250000000001,
        1802.8000000000002,
        1877.875,
        1952.95,
        2028.025,
        2103.1,
        2178.175,
        2253.25,
        2328.3250000000003,
        2403.4,
        2478.475,
        2553.55,
        2628.625,
        2703.7000000000003,
        2778.775,
        2853.85,
        2928.925,
        3004
      ],
      "coveragepos": [
        38.54,
        113.61,
        188.69,
        263.76,
        338.84,
        413.91,
        488.99,
        564.06,
```

```
        639.14,
        714.21,
        789.29,
        864.36,
        939.44,
        1014.51,
        1089.59,
        1164.66,
        1239.74,
        1314.81,
        1389.89,
        1464.96,
        1540.04,
        1615.11,
        1690.19,
        1765.26,
        1840.34,
        1915.41,
        1990.49,
        2065.56,
        2140.64,
        2215.71,
        2290.79,
        2365.86,
        2440.94,
        2516.01,
        2591.09,
        2666.16,
        2741.24,
        2816.31,
        2891.39,
        2966.46
      ],
      "width": [
        75.075,
        75.075,
        75.075,
        75.075,
        75.075,
        75.075,
        75.075,
        75.075,
        75.075,
        75.075,
        75.075,
        75.075,
        75.075,
        75.075,
        75.075,
        75.075,
        75.075,
        75.075,
        75.075,
        75.075,
        75.075,
        75.075,
        75.075,
        75.075,
        75.075,
        75.075,
        75.075,
        75.075,
        75.075,
        75.075,
        75.075,
        75.075,
        75.075,
        75.075,
        75.075,
        75.075,
        75.075,
        75.075,
        75.075,
        75.075
      ]
    },
    "percentile": {
      "Q1": 33,
      "Q2": 59,
      "Q3": 96
    },
    "max": 3004,
    "min": 0,
    "mean": 76.24469254060716,
    "median": 59,
    "zerocov": 46492724,
    "status": "ok",
    "warnthreshold": 40
  }
]
```


Caja de código A.5: Archivo en formato JSON percentile.json

```
{
  "warnthreshold": 0.3,
  "results": [
    {
      "bamfilename": "/home/agarcia/exomadata/hg00097.bam",
      "histdata": {
        "numberstd": [
          1,
          0,
          2,
          6,
          17,
          74,
          117,
          201,
          367,
          486,
          715,
          953,
          1193,
          1549,
          1741,
          2110,
          2369,
          2763,
          3022,
          3050,
          3214,
          3367,
          3575,
          3469,
          3492,
          3411,
          3557,
          3545,
          3444,
          3395,
          3366,
          3290,
          3358,
          3202,
          3244,
          3269,
          3170,
          3036,
          3009,
          3098,
          3066,
          3022,
          3053,
          3094,
          2992,
          3003,
          2919,
          3033,
          2900,
          2949,
          2890,
          2938,
          2764,
          2709,
          2753,
          2703,
          2534,
          2484,
          2420,
          2337,
          2202,
          2183,
          2151,
          1984,
          1918,
          1962,
          1755,
          1641,
          1672,
          1587,
          1431,
          1366,
          1206,
          1169,
          1144,
          1056,
          940,
          947,
          865,
          814,
          722,
          700,
          663,
          636,
          606,
          537,
          519,
          443,
          388,
          410,
          375,
          376,
```

```
            326,
            329,
            334,
            314,
            257,
            267,
            234,
            242,
            227,
            233,
            186,
            190,
            192,
            161,
            155,
            169,
            153,
            144,
            163,
            128,
            130,
            112,
            121,
            107,
            112,
            121,
            115,
            89,
            111,
            88,
            97,
            97,
            92,
            91,
            76,
            77,
            70,
            81,
            60,
            62,
            56,
            67,
            58,
            74,
            43,
            64,
            57,
            54,
            59,
            63
          ],
          "binedges": [
            0,
            0.007,
            0.014,
            0.021,
            0.028,
            0.035,
            0.042,
            0.049,
            0.056,
            0.063,
            0.07,
            0.077,
            0.084,
            0.091,
            0.098,
            0.105,
            0.112,
            0.11900000000000001,
            0.126,
            0.133,
            0.14,
            0.147,
            0.154,
            0.161,
            0.168,
            0.17500000000000002,
            0.182,
            0.189,
            0.196,
            0.203,
            0.21,
            0.217,
            0.224,
            0.231,
            0.23800000000000002,
            0.245,
            0.252,
            0.259,
            0.266,
            0.273,
            0.28,
            0.28700000000000003,
            0.294,
            0.301,
            0.308,
            0.315,
            0.322,
```

```
            0.329,
            0.336,
            0.343,
            0.35000000000000003,
            0.357,
            0.364,
            0.371,
            0.378,
            0.385,
            0.392,
            0.399,
            0.406,
            0.41300000000000003,
            0.42,
            0.427,
            0.434,
            0.441,
            0.448,
            0.455,
            0.462,
            0.46900000000000003,
            0.47600000000000003,
            0.483,
            0.49,
            0.497,
            0.504,
            0.511,
            0.518,
            0.525,
            0.532,
            0.539,
            0.546,
            0.553,
            0.56,
            0.5670000000000001,
            0.5740000000000001,
            0.581,
            0.588,
            0.595,
            0.602,
            0.609,
            0.616,
            0.623,
            0.63,
            0.637,
            0.644,
            0.651,
            0.658,
            0.665,
            0.672,
            0.679,
            0.686,
            0.6930000000000001,
            0.7000000000000001,
            0.707,
            0.714,
            0.721,
            0.728,
            0.735,
            0.742,
            0.749,
            0.756,
            0.763,
            0.77,
            0.777,
            0.784,
            0.791,
            0.798,
            0.805,
            0.812,
            0.8190000000000001,
            0.8260000000000001,
            0.833,
            0.84,
            0.847,
            0.854,
            0.861,
            0.868,
            0.875,
            0.882,
            0.889,
            0.896,
            0.903,
            0.91,
            0.917,
            0.924,
            0.931,
            0.9380000000000001,
            0.9450000000000001,
            0.9520000000000001,
            0.9590000000000001,
            0.966,
            0.973,
            0.98,
            0.987,
            0.994
        ],
        "stdpos": [
            0,
```

```
298             0.01,
299             0.02,
300             0.02,
301             0.03,
302             0.04,
303             0.05,
304             0.05,
305             0.06,
306             0.07,
307             0.07,
308             0.08,
309             0.09,
310             0.09,
311             0.1,
312             0.11,
313             0.12,
314             0.12,
315             0.13,
316             0.14,
317             0.14,
318             0.15,
319             0.16,
320             0.16,
321             0.17,
322             0.18,
323             0.19,
324             0.19,
325             0.2,
326             0.21,
327             0.21,
328             0.22,
329             0.23,
330             0.23,
331             0.24,
332             0.25,
333             0.26,
334             0.26,
335             0.27,
336             0.28,
337             0.28,
338             0.29,
339             0.3,
340             0.3,
341             0.31,
342             0.32,
343             0.33,
344             0.33,
345             0.34,
346             0.35,
347             0.35,
348             0.36,
349             0.37,
350             0.37,
351             0.38,
352             0.39,
353             0.4,
354             0.4,
355             0.41,
356             0.42,
357             0.42,
358             0.43,
359             0.44,
360             0.44,
361             0.45,
362             0.46,
363             0.47,
364             0.47,
365             0.48,
366             0.49,
367             0.49,
368             0.5,
369             0.51,
370             0.51,
371             0.52,
372             0.53,
373             0.54,
374             0.54,
375             0.55,
376             0.56,
377             0.56,
378             0.57,
379             0.58,
380             0.58,
381             0.59,
382             0.6,
383             0.61,
384             0.61,
385             0.62,
386             0.63,
387             0.63,
388             0.64,
389             0.65,
390             0.65,
391             0.66,
392             0.67,
393             0.68,
394             0.68,
395             0.69,
396             0.7,
```

```
        0.7,
        0.71,
        0.72,
        0.72,
        0.73,
        0.74,
        0.75,
        0.75,
        0.76,
        0.77,
        0.77,
        0.78,
        0.79,
        0.79,
        0.8,
        0.81,
        0.82,
        0.82,
        0.83,
        0.84,
        0.84,
        0.85,
        0.86,
        0.86,
        0.87,
        0.88,
        0.89,
        0.89,
        0.9,
        0.91,
        0.91,
        0.92,
        0.93,
        0.93,
        0.94,
        0.95,
        0.96,
        0.96,
        0.97,
        0.98,
        0.98,
        0.99
      ],
      "width": [
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
        0.007,
```

```
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007,
          0.007
        ]
      },
      "percentile": {
        "Q1": 0.1942488071712834,
        "Q2": 0.2976809691401779,
        "Q3": 0.4158891342754005
      },
      "max": 15.491933384829666,
      "min": 0,
      "mean": 0.3351479722152822,
      "median": 0.2976809691401779,
      "status": "warning"
```

```
595       }
596     ]
597 }
```

Caja de código A.6: Archivo en formato JSON std_wexons.json